Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2016, Article ID 9481971, 12 pages
http://dx.doi.org/10.1155/2016/9481971

Research Article

Generative Power and Closure Properties of Watson-Crick Grammars

Nurul Liyana Mohamad Zulkufli, Sherzod Turaev, Mohd Izzuddin Mohd Tamrin, and Azeddine Messikh

Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, 53100 Kuala Lumpur, Malaysia

Correspondence should be addressed to Nurul Liyana Mohamad Zulkufli; nurulliyanazulkufli@gmail.com

Received 27 November 2015; Revised 19 February 2016; Accepted 2 June 2016

Academic Editor: Ryotaro Kamimura

We define *WK linear grammars*, as an extension of WK regular grammars with linear grammar rules, and *WK context-free grammars*, thus investigating their computational power and closure properties. We show that WK linear grammars can generate some context-sensitive languages. Moreover, we demonstrate that the family of WK regular languages is the proper subset of the family of WK linear languages, but it is not comparable with the family of linear languages. We also establish that the Watson-Crick regular grammars are closed under almost all of the main closure operations.

1. Introduction

DNA computing appears as a challenge to design new types of computing devices, which differ from classical counterparts in fundamental way, to solve wide spectrum of computationally intractable problems. DNA (deoxyribonucleic acid) is double-stranded chain of nucleotides, which differ by their chemical bases that are *adenine* (A), *guanine* (G), *cytosine* (C), and *thymine* (T), and they are paired as A-T, C-G according to the *Watson-Crick complementary* as it is illustrated in Figure 1 [1]. The *massive parallelism*, another fundamental feature of DNA molecules, allows performing millions of cut and paste operations simultaneously on DNA strands until a complete set of new DNA strands performing are generated. These two features give high hope for the use of DNA molecules and DNA based biooperations to develop powerful computing paradigms and devices.

Since a DNA strand can be interpreted as a double strand sequence of symbols, the DNA replication and synthesize processes can be modeled using methods and techniques of formal language theory. *Watson-Crick (WK) automata* [2], one of the recent computational models abstracting the properties of DNA molecules, are finite automata with two reading heads, working on complete double-stranded sequences where characters on corresponding positions from the two strands of the input are related by a complementarity relation similar to the Watson-Crick complementarity of DNA nucleotides. The two strands of the input are separately read from left to right by heads controlled by a common state. Several variants have been introduced and studied in recent papers [3–7].

WK regular grammars [8], a grammar counterpart of WK automata, generate double-stranded strings related by a complementarity relation as in a WK automaton but use rules as in a regular grammar. The approach of using formal grammars in the study of biological and computational properties of DNA molecules by formal grammars is a new direction in the field of DNA computing: we can introduce powerful variants of WK grammars, such as WK linear, WK context-free, and WK regulated grammars, and use them in the investigation of the properties of DNA structures and also in DNA applications in food authentication, gene disease detection, and so forth. In this paper, we introduce *WK linear grammars* and study the generative capacity in the relationship of Chomsky grammars.

Further, as a motivation, we show synthesis processes, for instance, in DNA replication (Figure 2) can be simulated by derivations in WK grammars. The replication of DNA begins at the origin(s). The double strand then separated by proteins, producing bubble-like shape(s). The synthesis of new strands

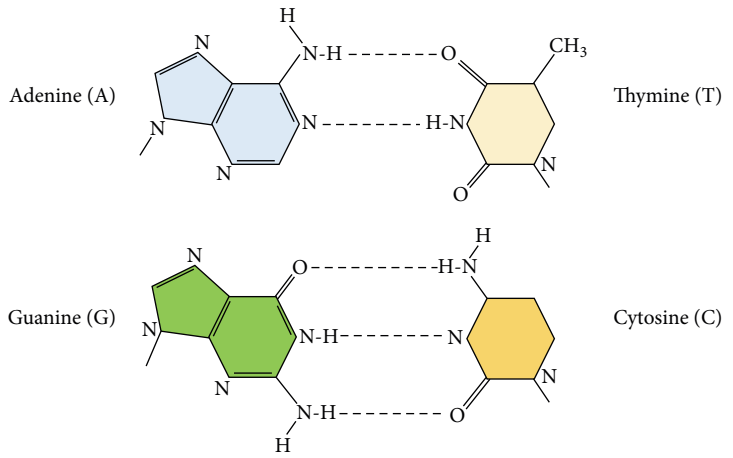

Figure 1: The structure of DNA strand.

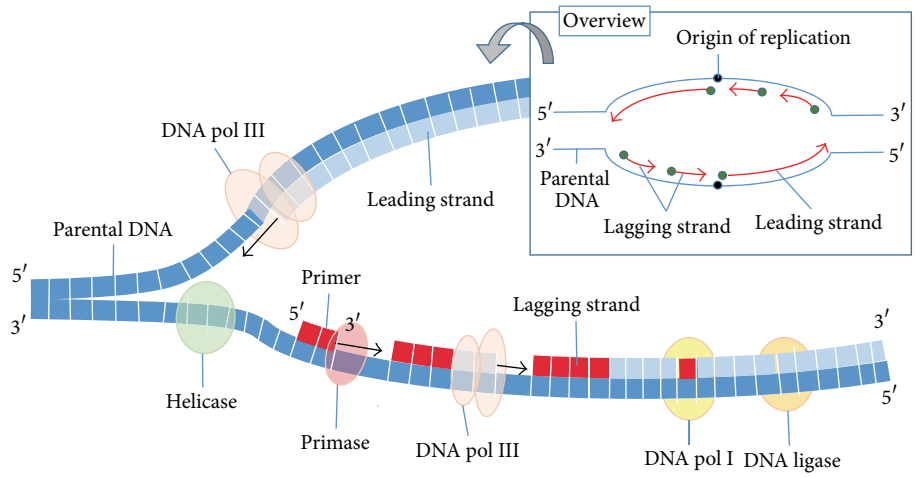

Figure 2: Synthesis process in bacterial DNA replication.

using the parental strands as templates starts from the origins and proceeds in the $5'$ to $3'$ direction of both strands [1].

This synthesis process in general can be seen as a *string generation*. The enzymes responsible for the synthesizing, DNA polymerases, cannot initiate the process by themselves but can only add nucleotides to an existing RNA chain. This chain is called *primer* which is produced by the enzyme primase. From the grammar perspective, the primase can be interpreted as the start symbol S. After the primer has been connected to the parental strand, one of the synthesizing enzymes, called DNA polymerase III, continues to add nucleotides one by one to the primer and are complemented with the parental strand. The synthesis finishes with replacing RNA primer with DNA nucleotides using the enzyme DNA polymerase I and joining with DNA ligase. Again, from the grammar perspective, DNA polymerase I and polymerase III act as production rules in the grammar, specially, DNA ligase resembles to a *terminal production* (see Figure 3).

The paper is organized as follows. In Section 2, we give some notions and definitions from the theories of formal languages and DNA computing needed in the sequel. In Section 3, we define WK grammars and languages generated

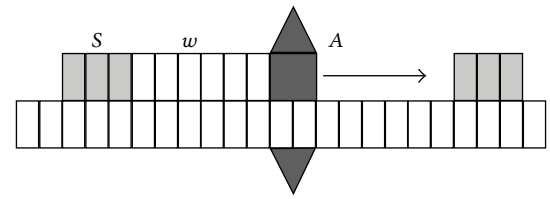

Figure 3: The simulation of a synthesis process with a derivation.

by these grammars. Section 4 is devoted to the study of the generative capacity of WK regular and linear grammars. In Section 5, we investigate the closure properties of WK grammars. Furthermore, we show the application of WK grammars in the analyses of DNA structures and programming language structures in Section 6. As the conclusion, we discuss open problems and interesting future research topics related to WK grammars in Section 7.

2. Preliminaries

We assume readers are familiar with formal languages theory and automata. Readers are referred to [3, 9–11].

Throughout this paper we use the following notations. Let $\in$ be the belonging relationship of an element to the corresponding set and $\notin$ indicates its negation. The symbol $\subseteq$ indicates the inclusion while $\subset$ notes the proper inclusion. The notations $\emptyset$, $|A|$, and 2^A denote the empty set, the cardinality of a set A, and the power set of A, respectively. When Σ is an alphabet (a finite set of symbols), the set of all finite strings is denoted by Σ^*, while Σ^+ shows similar meaning without including empty strings (we use λ for empty string). The length of a string $a \in \Sigma^*$ is shown by $|a|$. A language is a subset $L \subseteq \Sigma^*$.

Next we recall some terms regarding the closure properties of languages. The *union* of two languages, $L_1 \cup L_2$, is the set of strings including the elements contained in both sets of L_1 and L_2. The *concatenation* of two languages is yielded by lining two strings from both languages which is shown by

$$L_1 L_2 = \{ab \mid a \in L_1,\ b \in L_2\}. \tag{1}$$

The *Kleene-star closure* is the closure under the Kleene $*$ operation, the set of all possible strings in L including the empty string. The *mirror image* of a word $a = a_1 a_2 \cdots a_n$ is $a^R = a_n \cdots a_2 a_1$. For language L, its mirror image is

$$L^R = \left\{a^R \mid a \in L\right\}. \tag{2}$$

A *substitution* is a mapping $s : \Sigma^* \to T^*$ where $s(\lambda) = \lambda$ and $s(a_1, a_2) = s(a_1)s(a_2)$ for $a_1, a_2 \in \Sigma^*$. The substitution for a language $L \subseteq \Sigma^*$; that is, $s(L)$ is the union of $s(y)$, where $y \in L$. A substitution s is called *finite* if its length $|s(a)|$ is finite for each $a \in \Sigma$. A *morphism* is a substitution where its length is 1.

A *Chomsky grammar* is defined by $G = (N, T, S, P)$ where N is the set of *nonterminal* symbols and T is the set of *terminal* symbols, and $N \cap T = \emptyset$. $S \in N$ is the *start symbol* while $P \subseteq (N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$ is the set of *production rules*. We write $\alpha \to \beta$ indicating the rewriting process of the strings based on the production rules $(\alpha, \beta) \in P$. The term x *directly derives* y is written as $x \Rightarrow y$ when

$$\begin{aligned} x &= a_1 \alpha a_2, \\ y &= a_1 \beta a_2 \end{aligned} \tag{3}$$

for some production rules $\alpha \to \beta \in P$. A grammar generates a *language* defined by

$$L(G) = \{w \in T^* : S \Longrightarrow^* w\}. \tag{4}$$

According to the forms of production rules, grammars are classified as follows. A grammar $G = (N, T, S, P)$ is called

(i) *context-sensitive* if each production has the form $u_1 A u_2 \to u_1 u u_2$, where $A \in N$, $u_1, u_2, \in (N \cup T)^*$, and $u \in (N \cup T)^+$;

(ii) *context-free* if each production has the form $A \to u$, where $A \in N$ and $u \in (N \cup T)^*$;

(iii) *linear* if each production has the form $A \to u_1 B u_2$ or $A \to u$, where $A, B \in N$ and $u_1, u_2, u \in T^*$;

(iv) *right-linear* if each production has the form $A \to uB$ or $A \to u$, where $A, B \in N$ and $u \in T^*$;

(v) *left-linear* if each production has the form $A \to Bu$ or $A \to u$, where $A, B \in N$ and $u \in T^*$.

A right-linear and left-linear grammars are called *regular*.

The families of languages generated by these grammars are REG, LIN, CF, and CS, respectively. The families of recursive enumerable languages are denoted by RE while the families of finite languages are denoted by FIN. Thus the next relation holds [11].

Theorem 1 (Chomsky hierarchy). *Consider*

$$FIN \subset REG \subset LIN \subset CF \subset CS \subset RE. \tag{5}$$

We recall the definition of a finite automaton. A *finite automaton* (FA) is a quintuple $M = (Q, V, q_0, F, \delta)$, where Q is the set of *states*, $q_0 \in Q$ is the *initial state,* and $F \subseteq Q$ is set of *final states*. Meanwhile V is an alphabet and $\delta : Q \times V \to 2^Q$ is called the *transition function*. The set (language) of all strings accepted by M is denoted by $L(M)$. We denote the family of languages accepted by finite automata by FA. Then, FA = REG (see [11]).

Next, we cite some basic definitions and results of Watson-Crick automata.

The key feature of WK automata is the *symmetric relation* on an alphabet V; that is, $\rho \subseteq V \times V$. In this paper, for simplicity, we use the form $\langle u/v \rangle$ to mention the elements (u, v) in the set of all pairs of strings $V \times V$ (which we choose to write as $[V/V]$), and, instead of $V^* \times V^*$, we write $\langle V^*/V^* \rangle$.

Watson-Crick domain is the set of well-formed double-stranded strings (molecules)

$$\mathrm{WK}_\rho(V) = \left[\frac{V}{V}\right]_\rho^* = \left\{\left[\frac{a}{b}\right] : a, b \in V,\ (a, b) \in \rho\right\}. \tag{6}$$

$\mathrm{WK}_\rho^+(V)$ holds the similar meaning without including λ. We write

$$\left[\frac{a_1}{b_1}\right]\left[\frac{a_2}{b_2}\right]\cdots\left[\frac{a_n}{b_n}\right] \in \mathrm{WK}_\rho \tag{7}$$

as $[u/v]$ where the *upper strand* is $u = a_1 a_2 \cdots a_n$ and the *lower strand* is $v = b_1 b_2 \cdots b_n$. Note that when the elements in the upper strand are complemented and have the same length with the lower strand,

$$\left\langle \frac{u}{v} \right\rangle = \left[\frac{u}{v}\right]. \tag{8}$$

A Watson-Crick finite automaton (WKFA) is 6-tuple

$$M = (Q, V, q_0, F, \delta, \rho), \tag{9}$$

where Q, V, q_0, and F are the same as a FA. Meanwhile the transition function δ is

$$\delta : Q \times \left\langle \frac{V^*}{V^*} \right\rangle \longrightarrow 2^Q, \tag{10}$$

where $\delta(q, \langle u/v \rangle)$ is not an empty set only for finitely many triples $(q, u, v) \in Q \times V^* \times V^*$. Similar to FA, we can write the relation in transition function $q_2 \in \delta(q_1, \langle u/v \rangle)$ as a rewriting rule in grammars; that is,

$$q_1 \left\langle \frac{u}{v} \right\rangle \longrightarrow \left\langle \frac{u}{v} \right\rangle q_2. \tag{11}$$

We describe the reflexive and transitive closure of $\rightarrow$ as $\rightarrow^*$. The language accepted by a WKFA M is

$$L(M) = \left\{ u : \left[\frac{u}{v} \right] \in \mathrm{WK}_\rho(V),\ q_0 \left[\frac{u}{v} \right] \longrightarrow^* \left[\frac{u}{v} \right] q,\ \text{where } q \in F \right\}. \tag{12}$$

The family of languages accepted is indicated by WKFA. It is shown in [10, 12] that

$$\mathrm{REG} \subset \mathrm{WKFA} \subset \mathrm{CS}. \tag{13}$$

3. Definitions

In this section we slightly modified the definition of Watson-Crick regular grammars introduced in [8] in order to extend the concept to linear grammars and context-free grammars.

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

(i) *regular* if each production has the form

$$A \longrightarrow \left\langle \frac{u}{v} \right\rangle B \quad \text{or } A \longrightarrow \left\langle \frac{u}{v} \right\rangle, \tag{14}$$

where $A, B \in N$ and $\langle u/v \rangle \in \langle T^*/T^* \rangle$;

(ii) *linear* if each production has the form

$$A \longrightarrow \left\langle \frac{u_1}{v_1} \right\rangle B \left\langle \frac{u_2}{v_2} \right\rangle \quad \text{or } A \longrightarrow \left\langle \frac{u}{v} \right\rangle, \tag{15}$$

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

$$A \longrightarrow \alpha, \tag{16}$$

where $A \in N$ and $\alpha \in (N \cup (\langle T^*/T^* \rangle))^*$.

Definition 3. Let $G = (N, T, \rho, S, P)$ be a WK linear grammar. We say that $x \in (N \cup \langle T^*/T^* \rangle)^*$ directly derives $y \in (N \cup \langle T^*/T^* \rangle)^*$, denoted by $x \Rightarrow y$, iff

$$x = \left\langle \frac{u_1}{v_1} \right\rangle A \left\langle \frac{u_2}{v_2} \right\rangle,$$
$$y = \left\langle \frac{u_1}{v_1} \right\rangle \left\langle \frac{u_3}{v_3} \right\rangle B \left\langle \frac{u_4}{v_4} \right\rangle \left\langle \frac{u_2}{v_2} \right\rangle \tag{17}$$
$$\text{or } y = \left\langle \frac{u_1}{v_1} \right\rangle \left\langle \frac{u}{v} \right\rangle \left\langle \frac{u_2}{v_2} \right\rangle,$$

where $A, B \in N$, $u_i, v_i \in \langle T^*/T^* \rangle$, $i = 1, 2, 3, 4$, and

$$A \longrightarrow \left\langle \frac{u_3}{v_3} \right\rangle B \left\langle \frac{u_4}{v_4} \right\rangle,$$
$$A \longrightarrow \left\langle \frac{u}{v} \right\rangle \in P. \tag{18}$$

Definition 4. Let $G = (N, T, \rho, S, P)$ be a WK context-free grammar. We say that $x \in (N \cup \langle T^*/T^* \rangle)^*$ directly derives $y \in (N \cup \langle T^*/T^* \rangle)^*$, denoted by $x \Rightarrow y$, if and only if

$$x = \left\langle \frac{u_1}{v_1} \right\rangle A \left\langle \frac{u_2}{v_2} \right\rangle,$$
$$y = \left\langle \frac{u_1}{v_1} \right\rangle \alpha \left\langle \frac{u_2}{v_2} \right\rangle, \tag{19}$$

where $A, B \in N$, $u_i, v_i \in \langle T^*/T^* \rangle$, $i = 1, 2, 3, 4$, and $A \rightarrow \alpha \in P$.

Remark 5. We use a common notion "*Watson-Crick grammars*" referring to any type of WK grammars.

Definition 6. The language generated by a WK grammar is a quintuple G which is defined as

$$L(G) = \left\{ u : \left[\frac{u}{v} \right] \in \mathrm{WK}_\rho(T),\ S \Longrightarrow^* \left[\frac{u}{v} \right] \right\}. \tag{20}$$

4. Generative Power of Watson-Crick Grammars

In this section, we establish results regarding the computational power of WK grammars.

4.1. A Normal Form for Watson-Crick Linear Grammars. Next, we define 1-normal form for WK linear grammars and show that, for every WK linear grammar G, there is an equivalent WK linear grammar G' in the normal form; that is, $L(G) = L(G')$.

Definition 7. A linear WK grammar $G = (N, T, \rho, P, S)$ is said to be in the 1-normal form if each rule in P of the form

$$A \longrightarrow \left\langle \frac{u_1}{v_1} \right\rangle B \left\langle \frac{u_2}{v_2} \right\rangle \quad \text{or } A \longrightarrow \left\langle \frac{u_1}{v_1} \right\rangle, \tag{21}$$

where $|u_i| \leq 1$, $|v_i| \leq 1$, $i = 1, 2$, and $A, B \in N$.

Lemma 8. *For every WK linear grammar G, there exists an equivalent WK linear grammar G' in the 1-normal form.*

Proof. Let $G = (N, T, \rho, S, P)$ be a WK linear grammar. Let

$$A \longrightarrow \left\langle \frac{a_{11} \cdots a_{1m_1}}{b_{11} \cdots b_{1n_1}} \right\rangle B \left\langle \frac{a_{2m_2} \cdots a_{21}}{b_{2n_2} \cdots b_{21}} \right\rangle \tag{22}$$

be a production in P where $m_1 > 1$, $m_2 > 1$, $n_1 > 1$, or $n_2 > 1$. Without loss of generality, we assume that $m_1 \geq n_1$ and $m_2 \geq n_2$. Then, we define the following sequence of right-linear and left-linear production rules:

$$\begin{aligned}
A &\longrightarrow \left\langle \frac{a_{11}}{b_{11}} \right\rangle A^r_{11}, \\
A^r_{11} &\longrightarrow \left\langle \frac{a_{12}}{b_{12}} \right\rangle A^r_{12} \\
&\vdots \\
A^r_{1n_1-1} &\longrightarrow \left\langle \frac{a_{1n_1}}{b_{1n_1}} \right\rangle A^r_{1n_1}, \\
A^r_{1n_1} &\longrightarrow \left\langle \frac{a_{1n_1+1}}{\lambda} \right\rangle A^r_{1n_1+1} \\
&\vdots \\
A^r_{1m_1-1} &\longrightarrow \left\langle \frac{a_{1m_1}}{\lambda} \right\rangle A^r_{1m_1}, \\
A^r_{1m_1} &\longrightarrow A^r_{21} \left\langle \frac{a_{21}}{b_{21}} \right\rangle, \\
A^r_{21} &\longrightarrow A^r_{22} \left\langle \frac{a_{22}}{b_{22}} \right\rangle \\
&\vdots \\
A^r_{1n_2-1} &\longrightarrow A^r_{2n_2} \left\langle \frac{a_{2n_2}}{b_{2n_2}} \right\rangle, \\
A^r_{2n_2} &\longrightarrow A^r_{2n_2+1} \left\langle \frac{a_{2n_2+1}}{\lambda} \right\rangle \\
&\vdots \\
A^r_{2m_2-1} &\longrightarrow \left\langle \frac{a_{2m_2}}{\lambda} \right\rangle,
\end{aligned} \tag{23}$$

where A^r_{1j}, $1 \leq j \leq m_1$, and A^r_{2j}, $1 \leq j \leq m_2 - 1$, are new nonterminals.

Let

$$r : A \longrightarrow \left\langle \frac{a_1 a_2 \cdots a_m}{b_1 b_2 \cdots b_n} \right\rangle \in P, \tag{24}$$

where $m > 1$ or $n > 1$. Without loss of generality, we assume that $m \geq n$. Then, we define the following sequence of right-linear production rules:

$$\begin{aligned}
A &\longrightarrow \left\langle \frac{a_1}{b_1} \right\rangle A^r_1, \\
A^r_1 &\longrightarrow \left\langle \frac{a_2}{b_2} \right\rangle A^r_2, \\
A^r_{n-1} &\longrightarrow \left\langle \frac{a_n}{b_n} \right\rangle A^r_n, \\
A^r_n &\longrightarrow \left\langle \frac{a_{n+1}}{\lambda} \right\rangle A^r_{n+1} \\
&\vdots \\
A^r_{m-1} &\longrightarrow \left\langle \frac{a_m}{\lambda} \right\rangle,
\end{aligned} \tag{25}$$

where A^r_i, $1 \leq i \leq m - 1$, are new nonterminals.

We construct a WK linear grammar $G' = (N \cup N', T, \rho, S, P \cup P')$, where P' consists of productions defined above for each $A \rightarrow \langle u_1/v_1 \rangle B \langle u_2/v_2 \rangle \in P$ with $|u_1| > 1$, $|v_1| > 1$, $|u_2| > 1$, or $|v_2| > 1$ and $A \rightarrow \langle u/v \rangle \in P$ with $|u| > 1$ or $|v| > 1$. Then, it is not difficult to see that, in every derivation, productions in the form of (22) and (24) in G can be replaced by the sequences of productions (23) and (25) in G' and vice versa. Thus, $L(G) = L(G')$. □

4.2. The Generative Power. The following results immediately follow from the definition of WK grammars.

Lemma 9. *The following inclusions hold:*

$$\begin{aligned}
WKREG &\subseteq WKLIN, \\
LIN &\subseteq WKLIN, \\
CF &\subseteq WKCF.
\end{aligned} \tag{26}$$

Next, we show that WK grammars can generate non-context-free languages:

$$\begin{aligned}
&\left\{a^n c^n b^n : n \geq 1\right\}, \\
&\left\{a^n b^m c^n d^m : n, m \geq 1\right\}, \\
&\left\{wcw : w \in \{a, b\}^*\right\}.
\end{aligned} \tag{27}$$

Example 10. Let $G_1 = (\{S, A, B\}, \{a, b, c\}, \{(a, a), (b, b), (c, c)\}, P, S)$ be a WK linear grammar, where P consists of the rules

$$\begin{aligned} S &\longrightarrow \left\langle \frac{a}{\lambda} \right\rangle S \left\langle \frac{b}{\lambda} \right\rangle, \\ S &\longrightarrow \left\langle \frac{a}{\lambda} \right\rangle A \left\langle \frac{b}{\lambda} \right\rangle, \\ A &\longrightarrow \left\langle \frac{c}{a} \right\rangle A, \\ A &\longrightarrow \left\langle \frac{\lambda}{c} \right\rangle B \left\langle \frac{\lambda}{b} \right\rangle, \\ B &\longrightarrow \left\langle \frac{\lambda}{c} \right\rangle B \left\langle \frac{\lambda}{b} \right\rangle, \\ B &\longrightarrow \left\langle \frac{\lambda}{\lambda} \right\rangle. \end{aligned} \tag{28}$$

In general, we have the derivation

$$\begin{aligned} S &\Longrightarrow^* \left\langle \frac{a^{n-1}}{\lambda} \right\rangle S \left\langle \frac{b^{n-1}}{\lambda} \right\rangle \\ &\Longrightarrow \left\langle \frac{a^n}{\lambda} \right\rangle A \left\langle \frac{b^n}{\lambda} \right\rangle \\ &\Longrightarrow^* \left\langle \frac{a^n c^n}{a^n} \right\rangle A \left\langle \frac{b^n}{\lambda} \right\rangle \\ &\Longrightarrow \left\langle \frac{a^n c^n}{a^n c} \right\rangle B \left\langle \frac{b^n}{b} \right\rangle \\ &\Longrightarrow^* \left\langle \frac{a^n c^n}{a^n c^n} \right\rangle B \left\langle \frac{b^n}{b^n} \right\rangle \\ &\Longrightarrow \left[\frac{a^n c^n b^n}{a^n c^n b^n} \right]. \end{aligned} \tag{29}$$

Thus, G_1 generates the language $L(G_1) = \{a^n c^n b^n : n \geq 1\} \in$ CS − CF.

Example 11. Let

$$\begin{aligned} G_2 = (&\{S, A, B, C, D\}, \{a, b, c, d\}, \\ &\{(a, a), (b, b), (c, c), (d, d)\}, P, S) \end{aligned} \tag{30}$$

be a WK regular grammar, and P consists of the rules

$$\begin{aligned} S &\longrightarrow \left\langle \frac{a}{\lambda} \right\rangle S \mid \left\langle \frac{a}{\lambda} \right\rangle A, \\ A &\longrightarrow \left\langle \frac{b}{\lambda} \right\rangle A \mid \left\langle \frac{b}{\lambda} \right\rangle B, \\ B &\longrightarrow \left\langle \frac{c}{a} \right\rangle B \mid \left\langle \frac{c}{a} \right\rangle C, \\ C &\longrightarrow \left\langle \frac{d}{b} \right\rangle C \mid \left\langle \frac{d}{b} \right\rangle D, \\ D &\longrightarrow \left\langle \frac{\lambda}{c} \right\rangle D \mid \left\langle \frac{\lambda}{d} \right\rangle D \mid \left\langle \frac{\lambda}{\lambda} \right\rangle. \end{aligned} \tag{31}$$

Then, we have the following derivation for $n, m \geq 1$:

$$\begin{aligned} S &\Longrightarrow^* \left\langle \frac{a^{n-1}}{\lambda} \right\rangle S \Longrightarrow \left\langle \frac{a^n}{\lambda} \right\rangle A \\ &\Longrightarrow^* \left\langle \frac{a^n b^{m-1}}{\lambda} \right\rangle A \Longrightarrow \left\langle \frac{a^n b^m}{\lambda} \right\rangle B \\ &\Longrightarrow^* \left\langle \frac{a^n b^m c^{n-1}}{a^{n-1}} \right\rangle B \Longrightarrow \left\langle \frac{a^n b^m c^n}{a^n} \right\rangle C \\ &\Longrightarrow^* \left\langle \frac{a^n b^m c^n d^{m-1}}{a^n b^{m-1}} \right\rangle C \Longrightarrow \left\langle \frac{a^n b^m c^n d^m}{a^n b^m} \right\rangle D \\ &\Longrightarrow^* \left\langle \frac{a^n b^m c^n d^m}{a^n b^m c^n} \right\rangle D \\ &\Longrightarrow^* \left\langle \frac{a^n b^m c^n d^m}{a^n b^m c^n d^m} \right\rangle D \\ &\Longrightarrow \left[\frac{a^n b^m c^n d^m}{a^n b^m c^n d^m} \right]. \end{aligned} \tag{32}$$

Hence, $L(G_2) = \{a^n b^m c^n d^m : n, m \geq 1\} \in$ CS − CF.

Example 12. Let $G_3 = (\{S, A, B, C\}, \{a, b, c\}, \{(a, a), (b, b)\}, P, S)$ be a WK linear grammar with P consisting of the following rules:

$$\begin{aligned} S &\longrightarrow \left\langle \frac{a}{\lambda} \right\rangle S \mid \left\langle \frac{b}{\lambda} \right\rangle S \mid \left\langle \frac{c}{\lambda} \right\rangle A, \\ A &\longrightarrow \left\langle \frac{a}{a} \right\rangle A \mid \left\langle \frac{b}{b} \right\rangle A \mid \left\langle \frac{\lambda}{c} \right\rangle B, \\ B &\longrightarrow \left\langle \frac{\lambda}{a} \right\rangle B \mid \left\langle \frac{\lambda}{b} \right\rangle B \mid \left\langle \frac{\lambda}{\lambda} \right\rangle. \end{aligned} \tag{33}$$

By rules $S \to \langle a/\lambda \rangle$ and $S \to \langle b/\lambda \rangle$, we obtain a sentential form $\langle w/\lambda \rangle S$ where $w \in \{a, b\}^*$. The derivation is continued by only possible rule $S \to \langle c/\lambda \rangle A$ and we have $\langle wc/\lambda \rangle A$. Further, we can only apply rules $A \to \langle a/a \rangle A$ and $A \to \langle b/b \rangle A$. By the symmetric relation ρ, the derivation results in $\langle wcw/w \rangle A$. Then, we can only apply $A \to \langle \lambda/c \rangle B$ continuing with rules $B \to \langle \lambda/a \rangle B$ and $B \to \langle \lambda/b \rangle B$ and obtain $\langle wcw/wcw \rangle B$. Finally, by rule $B \to \langle \lambda/\lambda \rangle$, we get $[wcw/wcw]$. Illustratively,

$$\begin{aligned} S &\Longrightarrow^* \left\langle \frac{w}{\lambda} \right\rangle S \Longrightarrow \left\langle \frac{wc}{\lambda} \right\rangle A \\ &\Longrightarrow^* \left\langle \frac{wcw}{w} \right\rangle A \Longrightarrow \left\langle \frac{wcw}{wc} \right\rangle B \\ &\Longrightarrow^* \left\langle \frac{wcw}{wcw} \right\rangle B \Longrightarrow \left[\frac{wcw}{wcw} \right]. \end{aligned} \tag{34}$$

Thus, $L(G_3) = \{wcw : w \in \{a, b\}^*\} \in$ CS − CF.

The following theorem follows from Lemma 9 and Examples 10, 11, and 12.

Theorem 13. *The following inclusions hold:*

$$\begin{aligned} LIN &\subsetneq WKLIN, \\ WKREG - CF &\neq \emptyset, \\ WKLIN - CF &\neq \emptyset. \end{aligned} \tag{35}$$

The following example shows that some WK linear languages cannot be generated by WK regular grammars.

Lemma 14. *The following language is not a WK regular language:*

$$\begin{aligned} L_1 &= \{a^n b^m a^n : 2n \leq m \leq 3n\} \\ &\in WKLIN - WKREG. \end{aligned} \tag{36}$$

Proof. The language L_1 can be generated by the following WK linear grammar $G_4 = (\{S, A, B\}, \{a, b\}, \{(a, a), (b, b)\}, S, P)$, where P consists of the rules:

$$\begin{aligned} S &\longrightarrow \left\langle \frac{a}{\lambda} \right\rangle S \left\langle \frac{a}{a} \right\rangle \mid \left\langle \frac{a}{\lambda} \right\rangle A \left\langle \frac{a}{a} \right\rangle, \\ A &\longrightarrow \left\langle \frac{bb}{a} \right\rangle A \mid \left\langle \frac{bbb}{a} \right\rangle A \mid \left\langle \frac{\lambda}{b} \right\rangle B, \\ B &\longrightarrow \left\langle \frac{\lambda}{b} \right\rangle B \mid \left\langle \frac{\lambda}{\lambda} \right\rangle. \end{aligned} \tag{37}$$

It is not difficult to see that

$$\begin{aligned} S &\Longrightarrow^* \left\langle \frac{a^{n-1}}{\lambda} \right\rangle S \left\langle \frac{a^{n-1}}{a^{n-1}} \right\rangle \Longrightarrow \left\langle \frac{a^n}{\lambda} \right\rangle A \left\langle \frac{a^n}{a^n} \right\rangle \\ &\Longrightarrow^* \left\langle \frac{a^n b^m}{a^n} \right\rangle A \left\langle \frac{a^n}{a^n} \right\rangle \Longrightarrow \left\langle \frac{a^n b^m}{a^n b} \right\rangle B \left\langle \frac{a^n}{a^n} \right\rangle \\ &\Longrightarrow^* \left\langle \frac{a^n b^m}{a^n b^m} \right\rangle B \left\langle \frac{a^n}{a^n} \right\rangle \Longrightarrow \left[\frac{a^n b^m a^n}{a^n b^m a^n} \right], \end{aligned} \tag{38}$$

where $2n \leq m \leq 3n$.

Next, we show that $L_1 \notin$ WKREG.

We suppose, by contradiction, that L_1 can be generated by a WK regular grammar $G' = (N, \{a, b\}, \rho, S, P)$. Without loss of generality, we assume that G' is in 1-normal form. Then, for each rule $u \longrightarrow v$ in P, we have $u \in N$ and

$$\begin{aligned} v \in \Bigg\{ &\left\langle \frac{a}{\lambda} \right\rangle, \left\langle \frac{\lambda}{a} \right\rangle, \left\langle \frac{a}{a} \right\rangle, \left\langle \frac{b}{\lambda} \right\rangle, \left\langle \frac{\lambda}{b} \right\rangle, \left\langle \frac{b}{b} \right\rangle, \left\langle \frac{a}{b} \right\rangle, \\ &\left\langle \frac{b}{b} \right\rangle \Bigg\} \quad (N \cup \{\lambda\}). \end{aligned} \tag{39}$$

Let $w = a^r b^s a^r$ be a string in L_1 such that $r > |P|$. Then, the double-stranded sequence $[a^r b^s a^r / a^r b^s a^r]$ is generated by the grammar G'.

Case 1. In any derivation for this string, first b can occur in the upper (or lower) strand if a^r has already been generated in the upper (or lower) strand. Thus, we obtain two possible successful derivations:

$$\begin{aligned} S &\Longrightarrow^* \left\langle \frac{a^r b}{a^k} \right\rangle \\ \text{or } S &\Longrightarrow^* \left\langle \frac{a^r b}{a^r b} \right\rangle, \end{aligned} \tag{40}$$

where $k \leq r$. In the latter derivation in (40), we cannot control the number of occurrences of b; that is, the derivation may not be successful. In the former derivation in (40), using the second strand, we can generate b^s:

$$S \Longrightarrow^* \left\langle \frac{a^r b}{a^k} \right\rangle \Longrightarrow^* \left\langle \frac{a^r b^s}{a^r b^t} \right\rangle, \quad t \leq s. \tag{41}$$

Equation (41) is continued by generating a's in the first strand and we can use the second strand to control their number. Consider

$$S \Longrightarrow^* \left\langle \frac{a^r b}{a^k} \right\rangle \Longrightarrow^* \left\langle \frac{a^r b^s}{a^r b^t} \right\rangle \Longrightarrow^* \left\langle \frac{a^r b^s a^i}{a^r b^s a^j} \right\rangle, \tag{42}$$

and i is related to s. Since $2r \leq s \leq 3r$, generally, i is not the same as r for all derivations.

Case 2. We can control the number of a's after b's by using the second strand for a's before b's. In this case, the number of b's cannot be related to the number of a's:

$$S \Longrightarrow^* \left\langle \frac{a^r b^l a^r}{a^r} \right\rangle. \tag{43}$$

In both cases, we cannot control the number of b's and the number of a's after b's at the same time using WK regular rules. □

Since strings $a^n b^m a^n$ are palindrome strings for even m's, the language $\{ww^R : w \in \{a, b\}^*\}$ is not in WKREG; that is, we have the following.

Corollary 15. *The following holds:*

$$LIN - WKREG \neq \emptyset. \tag{44}$$

4.3. Hierarchy of the Families of Watson-Crick Languages. Combining the results above, we obtain the following theorem.

Theorem 16. *The relations in Figure 4 hold; the dotted lines denote incomparability of the language families and the arrows denote proper inclusions of the lower families into the upper families, while the dotted arrows denote inclusions.*

5. Closure Properties

In this section, we establish results regarding the closure properties of WK grammars. The families of WK languages are shown to be higher in the hierarchy than their respective Chomsky language families; thus it is interesting to see how

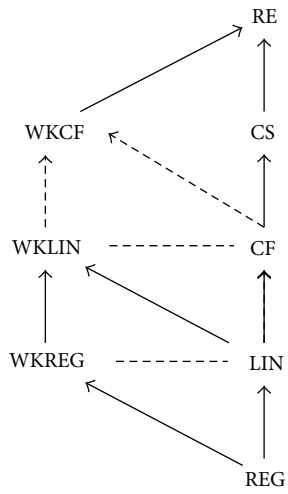

Figure 4: The hierarchy of WK and Chomsky language families.

WK grammars work in terms of closure properties as the ones for the Chomsky languages. Moreover, researching closure properties of WK grammars ensure the safety and correctness of the results yielded when performing operations on the sets of DNA molecules generated by some WK grammars.

5.1. Watson-Crick Regular Grammars. Let $L_1, L_2 \in$ WKREG, and $G_1 = (N_1, T, S_1, P_1, \rho)$, $G_2 = (N_2, T, S_2, P_2, \rho)$ be Watson-Crick regular grammars generating languages L_1 and L_2, respectively; that is, $L_1 = L(G_1)$ and $L_2 = L(G_2)$. Without loss of generality, we can assume that $N_1 \cap N_2 \neq \emptyset$ and S_1 and S_2 do not appear on the right-hand side of any production rule.

Lemma 17 (union). *WKREG is closed under union.*

Proof. Define $G = (N, T, S, P, \rho)$ by setting $N = N_1 \cup N_2 \cup \{S\}$ with $S \notin N_1 \cup N_2$, and

$$P = P_1 \cup P_2 \cup \{S \longrightarrow S_1\} \cup \{S \longrightarrow S_2\}. \tag{45}$$

Then it is not difficult to see that $L(G) = L_1 \cup L_2$. □

Lemma 18 (concatenation). *WKREG is closed under concatenation.*

Proof. Define $G = (N, T, S, P, \rho)$, where $N = N_1 \cup N_2$ and $S = S_1$. We define

$$\begin{aligned} P = P_2 \cup \left(P_1 - \left\{A \longrightarrow \left\langle \frac{u}{v} \right\rangle \in P_1\right\}\right) \\ \cup \left\{A \longrightarrow \left\langle \frac{u}{v} \right\rangle S_2 \mid A \longrightarrow \left\langle \frac{u}{v} \right\rangle \in P_1\right\}. \end{aligned} \tag{46}$$

Then it is obvious that $L(G) = L_1 \cdot L_2$. □

Lemma 19 (Kleene-star). *WKREG is closed under Kleene-star operation.*

Proof. Define the WK regular grammar $G = (N_1, T_1, S, P_1, \rho)$ with

$$P_1 = (P_1 - P_T) \cup P_S \cup \{S_1 \longrightarrow \lambda\}, \tag{47}$$

where

$$\begin{aligned} P_T &= \left\{A \longrightarrow \left\langle \frac{u}{v} \right\rangle \in P : \left\langle \frac{u}{v} \right\rangle \in \left\langle \frac{T^*}{T^*} \right\rangle\right\}, \\ P_S &= \left\{A \longrightarrow \left\langle \frac{u}{v} \right\rangle S_1 \mid A \longrightarrow \left\langle \frac{u}{v} \right\rangle \in P_T\right\}. \end{aligned} \tag{48}$$

Then, $L(G) = L_1^*$. □

Lemma 20 (finite substitution and homomorphism). *WKREG is closed under finite substitution and homomorphism.*

Proof. We show that the finite substitution (homomorphism) of L_1 is also in WKREG. Let $s : T^* \to \Sigma^*$ be a finite substitution (homomorphism). Define $G = (N_1, T_1, S_1, P, \rho_1)$, where $L(G) = s(L_1)$. Without loss of generality, we assume that G_1 is in the 1-normal form. Then, P_1 can contain production rules of the forms

$$\begin{aligned} &A \longrightarrow \left\langle \frac{a}{b} \right\rangle B, \\ &A \longrightarrow \left\langle \frac{a}{b} \right\rangle, \\ &A \longrightarrow \left\langle \frac{a}{\lambda} \right\rangle B, \\ &A \longrightarrow \left\langle \frac{a}{\lambda} \right\rangle, \\ &A \longrightarrow \left\langle \frac{\lambda}{b} \right\rangle B, \\ &A \longrightarrow \left\langle \frac{\lambda}{b} \right\rangle, \\ &A \longrightarrow \left\langle \frac{\lambda}{\lambda} \right\rangle. \end{aligned} \tag{49}$$

We define P as the set of production rules of the forms

$$\begin{aligned} &A \longrightarrow \left\langle \frac{s(a)}{s(b)} \right\rangle B, \\ &A \longrightarrow \left\langle \frac{s(a)}{s(b)} \right\rangle, \\ &A \longrightarrow \left\langle \frac{s(a)}{\lambda} \right\rangle B, \\ &A \longrightarrow \left\langle \frac{s(a)}{\lambda} \right\rangle, \\ &A \longrightarrow \left\langle \frac{\lambda}{s(b)} \right\rangle B, \end{aligned}$$

$$A \longrightarrow \left\langle \frac{\lambda}{s(b)} \right\rangle,$$
$$A \longrightarrow \left\langle \frac{\lambda}{\lambda} \right\rangle. \tag{50}$$

Since the substitution/homomorphism s is finite, P is a finite set too; that is, G is a WK regular grammar, and $L(G) = s(L_1)$. □

Lemma 21 (mirror image). *WKREG is closed under mirror image.*

Proof. We show that $L_1^R \in$ WKREG. Define $G = (N_1 \cup \{S\}, T_1, S, P, \rho_1)$ generating the language L_1^R, where S is a new nonterminal and P consists of production rules defined as follows:

(i) $A \to B\langle u/v\rangle$, where $A \to \langle u/v\rangle B \in P_1$,

(ii) $A \to \langle u/v\rangle$, where $A \to \langle u/v\rangle \in P_1$.

□

The results obtained from the lemmas above are summarized in the following theorem.

Theorem 22. *The family of Watson-Crick regular languages is closed under union, concatenation, Kleene-star, finite substitution, homomorphism, and mirror image.*

This shows that WK regular grammars preserve almost all of the closure properties of regular grammars. Other closure properties of WK regular grammars are left for future studies.

5.2. Watson-Crick Linear Grammars. Similar to the subsection above, a Watson-Crick linear grammar G is constructed for the purpose of investigating the closure properties of WKLIN.

Let $L_1, L_2 \in$ WKLIN, and

$$G_1 = (N_1, T, S_1, P_1, \rho),$$
$$G_2 = (N_2, T, S_2, P_2, \rho) \tag{51}$$

be Watson-Crick linear grammars generating L_1 and L_2, respectively; that is, $L_1 = L(G_1)$ and $L_2 = L(G_2)$. Without loss of generality, we can assume that $N_1 \cup N_2 \neq \emptyset$.

Lemma 23 (union). *WKLIN is closed under union.*

Proof. Define $G = (N, T, S, P, \rho)$, where $N = N_1 \cup N_2 \cup \{S\}$ with $S \notin N_1 \cup N_2$ and

$$P = P_1 \cup P_2 \cup \{S \longrightarrow S_1\} \cup \{S \longrightarrow S_2\}. \tag{52}$$

Then, $L(G) = L_1 \cup L_2$. □

Lemma 24 (homomorphism). *WKLIN is closed under homomorphism.*

Proof. Define $G = (N_1, T, S_1, P, \rho)$, where $L(G) = h(L_1)$. We show that the homomorphism of L_1 is also in WKLIN. Let $h : T^* \to \Sigma^*$ be a homomorphism. Without loss of generality, we assume that G_1 is in the 1-normal form. For each rule

$$r : A \longrightarrow \left\langle \frac{u_1}{v_1} \right\rangle B \left\langle \frac{u_2}{v_2} \right\rangle \in P_1, \tag{53}$$

where $u_1, v_1, u_2, v_2 \in T \cup \{\lambda\}$, we construct

$$h(r) : A \longrightarrow \left\langle \frac{h(u_1)}{h(v_1)} \right\rangle B \left\langle \frac{h(u_2)}{h(v_2)} \right\rangle \tag{54}$$

in P.

In every successful derivation of G_1 generating $[w/w] \in [T/T]^*$, we replace production rule $r \in P_1$ with the production rule $h(r) \in P$ and obtain the string $[h(w)/h(w)] \in [\Sigma/\Sigma]^*$. Thus, $h(w) \in h(L_1)$. □

Theorem 25. *The family of Watson-Crick linear languages is closed under union and homomorphism.*

It is compelling to prove if the concatenation of two WK linear grammars is still included in WKLIN or not. This also decides whether WK linear grammars are unique from linear grammars or not, as linear grammars are not closed under concatenation and thus not closed under Kleene-star operation.

5.3. Watson-Crick Context-Free Grammars. Let $L_1, L_2, L \in$ WKCF, and $G_1 = (N_1, T, S_1, P_1, \rho), G_2 = (N_2, T, S_2, P_2, \rho)$ be Watson-Crick context-free grammars generating L_1 and L_2, respectively; that is, $L_1 = L(G_1)$ and $L_2 = L(G_2)$. Without loss of generality, we can assume that $N_1 \cup N_2 \neq \emptyset$.

Lemma 26 (union). *WKCF is closed under union operation.*

Proof. Define the WK context-free grammar $G = (N, T, S, P, \rho)$, where $N = N_1 \cup N_2 \cup \{S\}$ with $S \notin N_1 \cap N_2$, and

$$P = P_1 \cup P_2 \cup \{S \longrightarrow S_1\} \cup \{S \longrightarrow S_2\}. \tag{55}$$

Then it is obvious that $L(G) = L_1 \cup L_2$. □

Lemma 27 (concatenation). *WKCF is closed under concatenation.*

Proof. Define $G = (N_1 \cup N_2 \cup \{S\}, T_1 \cup T_2, S, P, \rho)$ where $S \notin (N_1 \cup N_2)$, and

$$P = P_1 \cup P_2 \cup \{S \longrightarrow S_1 S_2\}. \tag{56}$$

Then, $L(G) \in$ WKCF. □

Lemma 28 (Kleene-star). *WKCF is closed under Kleene-star operation.*

Proof. Define $G = (N_1 \cup \{S\}, T_1, S, P, \rho) \in$ WKCF setting

$$P = P_1 \cup \{S \longrightarrow S_1 S \mid \lambda\}. \tag{57}$$

Then it is not difficult to see that $L(G) \in$ WKCF. □

Lemma 29 (homomorphism). *WKCF is closed under homomorphism.*

Proof. Define $G = (N_1, T, S_1, P, \rho)$, where $L(G) = h(L_1)$. We show that the homomorphism of L_1 is also in WKCF. Let $s : T^* \to \Sigma^*$ be a homomorphism. P_1 contains production rules of the form

$$r : A \longrightarrow \left\langle \frac{u_1}{v_1} \right\rangle B_1 \left\langle \frac{u_2}{v_2} \right\rangle B_2 \cdots \left\langle \frac{u_k}{v_k} \right\rangle B_k \left\langle \frac{u_{k+1}}{v_{k+1}} \right\rangle, \tag{58}$$

where $k \geq 0$.

For each rule $r \in P$, we construct

$$h(r) : A \longrightarrow \left\langle \frac{h(u_1)}{h(v_1)} \right\rangle B_1 \left\langle \frac{h(u_2)}{h(v_2)} \right\rangle B_2 \cdots \left\langle \frac{h(u_k)}{h(v_k)} \right\rangle B_k \left\langle \frac{h(u_{k+1})}{h(v_{k+1})} \right\rangle, \tag{59}$$

where $k \geq 0$.

In every successful derivation of G_1 generating $[w/w] \in [T/T]^*$, we replace production rule $r \in P_1$ with the production rule $h(r) \in P$ and obtain the string $[h(w)/h(w)] \in [\Sigma/\Sigma]^*$. Thus, $h(w) \in h(L_1)$. □

With the lemmas provided above in this subsection, the next theorem follows.

Theorem 30. *The family of Watson-Crick context-free languages is closed under union, concatenation, Kleene-star, and homomorphism.*

Closure of WKCF under complement and intersection depends on the generative capacity of WKCF. The nonclosure of context-free (CF) grammars for intersection was shown with the famous example that the intersection of two CF languages results in a string that cannot be generated by a CF grammars. If one can provide some examples of strings that cannot be generated by WK context-free grammars, then the nonclosure of WKCF can be proven.

6. Applications of Watson-Crick Grammars

In this section, we consider two examples of the applications of Watson-Crick grammars in the analyses of DNA structures and programming language structures.

6.1. DNA Structure Analysis. Since Watson-Crick grammars are developed based on the structure and recombinant behavior of DNA molecules, they can suitably be implemented in the study of DNA related problems.

The analysis of DNA strings provides useful information: for instance, the finding of a specific pattern in a DNA string and the identification of the repeats of a pattern are very important for detecting mutation. One of the diseases caused by mutation is Huntington disease, resulting from trinucleotide repeat disorders [13–15]. It is discovered that the number of repeats of the trinucleotide *CTG*/*CAG* in the patient's DNA with Huntington disease is not normal. The repeats are also useful for finding the origin of replication of microorganisms [16].

In this section, we show that Watson-Crick grammars can be used for analyzing the repeats in DNA strings. We use the DNA of a breed of pig, Sus scrofa breed mixed chromosome 1, Sscrofa10.2 provided by the The National Center for Biotechnology Information (NCBI) database (NCBI Reference Sequence: NC_010443.4) [17].

Consider a part of the upper strand of the DNA from the breed stated above with the length of 100 nucleotides:

$$\begin{aligned} &aatcgacgcc\ acacg\mathbf{cag}gc\ \mathbf{cag}ttccgag\ \mathbf{ctg}cat\mathbf{ctg}g\ gac\mathbf{ctg}ctcc \\ &atat\mathbf{ctg}aag\ gcaatc\mathbf{ctg}g\ atc\mathbf{cag}ggat\ tgcac\mathbf{ctg}cc\ ttctctccta. \end{aligned} \tag{60}$$

In this example, the pattern *ctg* is being repeated for six times in the direct strand (upper strand) and two times in the reverse strand (lower strand) which can be seen as *cag* in the upper strand. Focusing on the repeats of *ctg* pattern, a DNA string containing such a pattern can be expressed as

$$\left(\{a,t,c,g\}^* ctg \{a,t,c,g\}^*\right)^*. \tag{61}$$

We now construct a simple Watson-Crick regular grammar G to generate the above language.

Let $G = (N, T, S, P, \rho)$, where $T = \{a, t, c, g\}$, $\rho = \{(a,t), (c,g)\}$, and P consists of the following productions:

$$\begin{aligned} &S \longrightarrow \left\langle \frac{a}{t} \right\rangle S \mid \left\langle \frac{t}{a} \right\rangle S \mid \left\langle \frac{g}{c} \right\rangle S \mid \left\langle \frac{c}{g} \right\rangle A, \\ &A \longrightarrow \left\langle \frac{c}{g} \right\rangle A \mid \left\langle \frac{a}{t} \right\rangle S \mid \left\langle \frac{g}{c} \right\rangle S \mid \left\langle \frac{t}{a} \right\rangle B, \\ &B \longrightarrow \left\langle \frac{c}{g} \right\rangle A \mid \left\langle \frac{a}{t} \right\rangle S \mid \left\langle \frac{t}{a} \right\rangle S \mid \left\langle \frac{g}{c} \right\rangle C, \\ &C \longrightarrow \left\langle \frac{a}{t} \right\rangle C \mid \left\langle \frac{t}{a} \right\rangle C \mid \left\langle \frac{g}{c} \right\rangle C \mid \left\langle \frac{c}{g} \right\rangle A \mid \left\langle \frac{\lambda}{\lambda} \right\rangle. \end{aligned} \tag{62}$$

For instance, a derivation resulting in six repeats of *ctg* pattern can be obtained as follows:

$$\begin{aligned} S &\Longrightarrow^* \left\langle \frac{x}{y} \right\rangle S \Longrightarrow \left\langle \frac{xc}{yg} \right\rangle A \Longrightarrow \left\langle \frac{xct}{yga} \right\rangle B \\ &\Longrightarrow \left\langle \frac{xctg}{ygac} \right\rangle C \\ &\Longrightarrow^* \left\langle \frac{xctgx}{ygacy} \right\rangle A \Longrightarrow \left\langle \frac{xctgxc}{ygacyg} \right\rangle A \\ &\Longrightarrow \left\langle \frac{xctgxct}{ygacyga} \right\rangle B \\ &\Longrightarrow \left\langle \frac{xctgxctg}{ygacygac} \right\rangle C \Longrightarrow^* \left\langle \frac{xctgxctgx}{ygacygacy} \right\rangle C \\ &\Longrightarrow^* \left[\frac{xctgxctgxctgxctgxctgxctgx}{ygacygacygacygacygacygacy} \right], \end{aligned} \tag{63}$$

where $x \in \{a, t, g\}^*$ and y is their complementarity symbols based on ρ, respectively.

Though, in this example, the functionality of WK regular grammars is not used to the fullest, the DNA structures can be naturally and effectively analyzed with WK grammars.

6.2. Programming Language Structure Analysis. The ability to differentiate between parentheses that are correctly balanced and those that are unbalanced is an important part of recognizing many programming language structures. Balanced parentheses mean that each opening symbol has a corresponding closing symbol and the pairs of parentheses are properly nested.

The parsing algorithms of compilers and interpreters have to check the correctness of balanced parentheses in the blocks of codes including algebraic and arithmetic expressions.

Although context-free grammars are used to develop parsers for programming languages, many programming language structures are context-sensitive. Thus, it is of interest to develop parsers based on grammars which are able to analyze non-context-free language structures.

Further, we show an example of balanced parentheses that can be generated by WK regular grammars but cannot be generated by context-free grammars. One can see that just by incorporating the concept of double-stranded string bonded with Watson-Crick complementarity, even WK grammars that are based of just regular rules can enhance the power of the parsing techniques.

To avoid confusion, we denote "(" as the open parenthesis terminal symbol and ")" as the close parenthesis terminal symbol in bold font.

Example 31. Let $G_5 = (\{S, A, B\}, \{(,)\}, \{((,))\}, S, P_5)$ be a WK regular grammar. P_5 consists of the rules

$$\begin{aligned}
S &\longrightarrow \left\langle \frac{(}{\lambda} \right\rangle S \\
S &\longrightarrow \left\langle \frac{(}{\lambda} \right\rangle A, \\
A &\longrightarrow \left\langle \frac{)}{(} \right\rangle A \\
A &\longrightarrow \left\langle \frac{)}{(} \right\rangle B \\
B &\longrightarrow \left\langle \frac{(}{)} \right\rangle B \\
B &\longrightarrow \left\langle \frac{\lambda}{)} \right\rangle B \\
B &\longrightarrow \left\langle \frac{\lambda}{\lambda} \right\rangle \\
B &\longrightarrow A.
\end{aligned} \tag{64}$$

From this, we obtain the derivation

$$\begin{aligned}
S &\Longrightarrow \left\langle \frac{(}{\lambda} \right\rangle S \Longrightarrow^* \left\langle \frac{(^n}{\lambda} \right\rangle A \\
&\Longrightarrow \left\langle \frac{(^n)}{(} \right\rangle A \Longrightarrow^* \left\langle \frac{(^n)^n}{(^n} \right\rangle B \\
&\Longrightarrow \left\langle \frac{(^n)^n}{(^n)} \right\rangle B \Longrightarrow^* \left\langle \frac{(^n)^n}{(^n)^n} \right\rangle B \\
&\Longrightarrow \left\langle \frac{(^n)^n(}{(^n)^n} \right\rangle B \Longrightarrow^* \left\langle \frac{(^n)^n(^n}{(^n)^n} \right\rangle A \\
&\Longrightarrow \left\langle \frac{(^n)^n(^n)}{(^n)^n} \right\rangle A \Longrightarrow^* \left\langle \frac{(^n)^n(^n)^n}{(^n)^n} \right\rangle B \\
&\Longrightarrow^* \cdots \Longrightarrow^* \left[\frac{((^n)^n)^k}{((^n)^n)^k} \right],
\end{aligned} \tag{65}$$

where $k \geq 1$. Hence, the language obtained is

$$L_5 = \left\{ \left((^n)^n \right)^k : n, k \geq 1 \right\} \in \text{WKREG} - \text{CF}. \tag{66}$$

7. Conclusion

In this paper, we defined Watson-Crick regular grammars, Watson-Crick linear grammars, and Watson-Crick context-free grammars. Further, we investigated their computational power and closure properties. We showed that

(1) WK linear grammars can generate some context-sensitive languages;

(2) the families of linear languages and WK regular languages are strictly included in the family of WK linear grammars;

(3) the families of WK regular languages and linear languages are not comparable;

(4) the family of WK linear languages is not comparable with the family of context-free languages;

(5) WK regular grammars preserves the closure properties similar to the ones of regular languages.

The following problems related to the topic remain open:

(1) Are the family of context-free languages proper subset of the family of WK linear languages or are they not comparable?

(2) What is the generative capacity of Watson-Crick context-free grammars?

(3) What are the remaining closure properties of Watson-Crick (regular, linear, and context-free) grammars? These depend on their generative capacity.

Moreover, there are many interesting topics for further research; for instance, we can define WK context-free and regulated WK context-free grammars and use them in the study of DNA properties and in the DNA based applications such as food authentication and disease gene detection.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work has been supported through International Islamic University Endowment B research Grant EDW B14-136-1021 and Fundamental Research Grant Scheme FRGS13-066-0307, Ministry of Education, Malaysia. The first author would like to thank both organizations for the scholarship through the IIUM fellowship program.

References

[1] J. B. Reece, L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, and R. B. Jackson, *Campbell Biology*, Pearson Education, 10th edition, 2011.

[2] R. Freund, G. Paun, G. Rozernberg, and A. Salomaa, *Watson-Crick Finite Automata*, vol. 48 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1999.

[3] E. Czeizler, "A short survey on watson-crick automata," *Bulletin of the EATCS*, vol. 88, no. 3, pp. 104–119, 2006.

[4] L. Kari, S. Seki, and P. Sosik, "DNA computing—foundations and implications," in *Handbook of Natural Computing*, G. Rozenberg, T. Bäck, and J. N. Kok, Eds., pp. 1073–1127, 2012.

[5] P. Leupold and B. Nagy, "$5' \to 3'$ Watson-Crick automata with several runs," *Fundamenta Informaticae*, vol. 104, no. 1-2, pp. 71–91, 2010.

[6] M. I. Mohd Tamrin, S. Turaev, and T. M. Tengku Sembok, "Weighted watson-crick automata," in *Proceedings of the 21st National Symposium on Methematical Sciences*, vol. 1605 of *AIP Conference Proceedings*, p. 302, Penang, Malaysia, November 2013.

[7] K. G. Subramanian, I. Venkat, and K. Mahalingam, "Context-free systems with a complementarity relation," in *Proceedings of the 6th International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA '11)*, pp. 194–198, Penang, Malaysia, September 2011.

[8] K. G. Subramanian, S. Hemalatha, and I. Venkat, "On Watson-Crick automata," in *Proceedings of the 2nd International Conference on Computer Science, Science, Engineering and Information Technology (CCSEIT '12)*, pp. 151–156, Coimbatore, India, 2012.

[9] P. Linz, *An Introduction to Formal Languages and Automata*, Jones and Bartlett, 2006.

[10] G. Păun, G. Rozenberg, and A. Salomaa, *DNA Computing*, New Computing Paradigms, Springer, Berlin, Germany, 1998.

[11] G. Rozenberg and A. Salomaa, *Handbook of Formal Languages*, vol. 1–3, Springer, 1997.

[12] S. Okawa and S. Hirose, "The relations among Watson-Crick automata and their relations with context-free languages," *IEICE Transactions on Information and Systems*, vol. E89, no. 10, pp. 2591–2599, 2006.

[13] M. E. MacDonald, C. M. Ambrose, M. P. Duyao et al., "A novel gene containing a trinucleotide repeat that is expanded and unstable on Huntington's disease chromosomes," *Cell*, vol. 72, no. 6, pp. 971–983, 1993.

[14] H. T. Orr and H. Y. Zoghbi, "Trinucleotide repeat disorders," *Annual Review of Neuroscience*, vol. 30, pp. 575–621, 2007.

[15] J. Petruska, M. J. Hartenstine, and M. F. Goodman, "Analysis of strand slippage in DNA polymerase expansions of CAG/CTG triplet repeats associated with neurodegenerative disease," *The Journal of Biological Chemistry*, vol. 273, no. 9, pp. 5204–5210, 1998.

[16] N. C. Jones and P. Pevzner, *An Introduction to Bioinformatics Algorithms*, MIT Press, Boston, Mass, USA, 2004.

[17] M. A. M. Groenen, A. L. Archibald, H. Uenishi et al., "Analyses of pig genomes provide insight into porcine demography and evolution," *Nature*, vol. 491, no. 7424, pp. 393–398, 2012.