Tree Valence Controlled Grammars

Salbiah Ashaari[1], Sherzod Turaev[1], Abdurahim Okhunov[2]
[1] *Department of Computer Science, Kulliyyah of Information and Communication Technology*
International Islamic University Malaysia, Kuala Lumpur, Malaysia
salbiah.ash@gmail.com, sherzod@iium.edu.my
[2] *Department of Science in Engineering, Kulliyyah of Engineering*
International Islamic University Malaysia, Kuala Lumpur, Malaysia
abdurahimokhun@iium.edu.my

Abstract— Beyond a shadow of a doubt, the studying of context-free grammars with restricted derivation trees known as tree controlled grammars have achieved plentiful remarkable results within formal language theory as demonstrated in a number of publications on this subject for the past forty five years. In principle, these grammars generate their languages as an ordinary context-free grammar except their derivation trees need to be satisfied by certain prescribed conditions. Our paper is a continuing of studying of this kind of grammars, where we introduce a new variant of tree controlled grammar called a *tree valence controlled grammar*, which replaces regular sets with valences where every main production with certain integer value will be derived into sub-productions with the value of combination of zero and one or zero and minus represented in matrices form with the permutations of each matrix row yield a zero value (at every level of tree derivation, the summation of valence value is zero). We also investigate the generative capacity and structural properties of these grammars.

Keywords— Context-Free Grammars, Tree Controlled Grammars, Generative Powers, Closure Properties.

I. INTRODUCTION

The idea of imposing restrictions upon the derivation trees of context-free grammars was originated by Culik and Maurer in 1977, when they introduced a new regulated grammar called tree controlled grammars (for short TC grammars). They defined TC grammars as a pair (G, R) with G is a context-free grammar and R is a regular language that is used as a control mechanism in generating a successful derivation in G. Then, a word, w that generated by G, is said to belong to the language designated by G and R if there exists a derivation tree t in G for w such that all tree levels except the last one are in R. The authors also studied the complexity of parsing of TC grammar where they found that it can be parsed in time $O(n^2)$ for both context free and some non-context free languages. At the same time, they proved that classes of languages of regular, linear, context free, recursively enumerable, EOL and ETOL can be characterized by TC grammars in an innate manner [1].

Since TC grammars were properly defined, a great extent of researches has been done on them in variety directions. In 1979, Paun thoroughly examined the computational power of TC grammars by not only considering the context-free grammars controlled by regular languages but also by considering all possible variants of the TC grammars with varying the grammars as well as their control languages. He studied the computational power of TC regular, context-free and λ-free context-free grammars controlled by all types of Chomsky languages including finite languages. Thus, he came out with fifteen types of TC grammars. In general, he proved that if erasing rules are prohibited in the productions, the TC grammars coincide with context-sensitive grammars and if erasing rules are allowed, they coincide with arbitrary phrase structure grammars [2].

Then, after around twenty years elapsed, there arose an issue whether there is a possibility for TC grammars to possess the same power if the underlying grammars are controlled by subregular languages. This issue was investigated by Dassow and Truthe in 2008 where they considered several different types of subregular languages such as regular suffix-closed languages, regular commutative languages, finite languages, circular languages, non-counting languages, nilpotent languages, ordered languages and combinational. In their study, they noticed subregularly TC grammars generate the context-sensitive, EOL and matrix languages [3]. Further, they continued to study on the hierarchy of such grammars where they presented several ideas of controlling derivation trees levels of context-free grammar by the regular languages with restricted complexity, by finite union of monoids and by languages accepting deterministic finite automaton (DFA) with mostly prescribed number of

states. As a result, they proved that at level 2, the corresponding hierarchy of TC languages already collapsed [4].

A specific research on the complexities of TC grammars was first conducted by Stiebe in 2008 where he validated that every linearly bounded queue automaton has a TC grammar. Then, he showed that without erasing productions, context-sensitive languages can be generated by a TC context-free grammar that has a control language accepted by deterministic finite automaton with at most five states but if erasing productions are allowed in the grammar, it can generate the recursively enumerable languages [5]. Subsequently, in 2011, Turaev, Dassow and Selamat recommenced to look into the TC grammars in the company of bounded nonterminal complexity where then they proved that without erasing rules, the nonterminals number in TC grammars can lead to an infinite hierarchy of TC languages families and with erasing rules, any recursively enumerable languages can be generated with no more than nine nonterminals [6]. The same authors again, Turaev et al. but in another paper [7], demonstrated that any recursively enumerable language can be generated by a TC grammar with at most seven nonterminals only. They also established that a TC grammar with three nonterminals is already sufficient to generate any regular simple matrix and linear languages. However, those two proofs are still undecidable whether they are optimal or not [7]. Interestingly, one year later, Turaev et al. again came back with another paper that investigated the same issue in [7] but this time they completely proved that the bound for the mentioned families of languages before are optimal. In addition, they presented that TC grammars with at most four nonterminals are sufficient enough to generate any context-free languages [8].

Later on but still in the same year, by using the same technique as done in [8] but with different version of the Geffert normal form, Vaszil demonstrated that the complexity of nonterminal of TC grammars can be reduced from seven to six [9].

In 2012, Koutny and Meduna came out with a new idea of generating TC grammars where instead of placing the restrictions on tree levels, they placed them on the tree paths and cuts. They restricted the derivation tree cuts by an advocated regular language with the notion that in every derivation tree in the grammar, there exists a set X of tree cuts which specified by regular language and cover all the tree. They showed that these grammars can characterize the family of languages of recursively enumerable. Not only that, they as well introduced a binary relation over those grammars together with the proof that it also can generate the identical family of languages of recursively enumerable [10].

This paper is structured as follows. First, we recall some basic notations, notions and concepts that will be used throughout this paper such formal languages theories, tree structure and grammars with regulated rewriting. Then, we introduce a new type of tree controlled grammars called tree valence controlled grammars. Further, we investigate its generative power as well as their closure properties. Lastly, we provide a brief summarization of all materials discussed in this paper.

II. PRELIMINARIES

In this section, we shortly recall the necessary basic notations, terminologies and concepts related to the formal languages theories, tree structure and regulated rewriting grammars that will be used in the following sections. Then, for more exhaustive information or unexplained notions, the reader can refer to [11] - [15].

All the way through this paper, we use the following basic notations. The symbols $\in$ and $\notin$ represent the set membership and negation of set membership of an element to a set. The symbol $\subseteq$ signifies the inclusion which is not necessarily proper and $\subset$ stands for the strict inclusion. Alternatively, we can use the symbol $\subsetneq$, i.e., for a two sets A and B, $A \subsetneq B$ if $A \subseteq B$ and $A \neq B$. Further, we have the notation $|A|$ to portray the cardinality of a set A which is the number of elements in the set A with 2^A to depict the power set of a set A. The symbol $\emptyset$ denotes the empty set which implies that there is no element in the set. Subsequently, the set of integer, natural, real and rational number are denoted by $\mathbb{Z}, \mathbb{N}, \mathbb{R}$ and $\mathbb{Q}$, respectively.

Afterwards, we have an alphabet, which is a finite and nonempty set of elements known as symbols or letters, denoted by Σ and a string (sometimes referred as word) over Σ which is a finite sequence of symbols (concatenation of symbols) from Σ. The string without symbols is called null or empty string and it is denoted by λ. The set of all strings (including λ) over the alphabet Σ is represented by Σ^*, and Σ^+ denotes $\Sigma^* - \{\lambda\}$. A string w is a substring of a string v if and only if there exist u_1, u_2 such that $v = u_1 w u_2$, $u_1, u_2, w, v \in \Sigma^*$. The length of string is denoted by $|w|$ that is the number of symbols in it the string. A language L is a subset of Σ^*. $L \subseteq \Sigma^*$ is λ -free if $\lambda \notin L$.

A context-free grammar is a quadruple $G = (N, T, S, P)$ where N and T are finite sets of nonterminal and terminal symbols, respectively, $S \in N$ is the start symbol and $P \in N \times (N \cup T)^*$ is the set of (production) rules. Usually, a rule (A, w) is written as $A \to w$. A rule of the form $A \to \lambda$ is called an erasing rule. $u \in (N \cup T)^+$ directly derives $v \in (N \cup T)^*$, written as $x \Rightarrow y$, if and only if there is a rule $r = A \to w \in P$ such that $u = u_1 A u_2$ and $v = v_1 w v_2$. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$. A derivation using the sequence of rules $\pi = r_1 r_2 \ldots r_n$ is

denoted by $\overset{\pi}{\Rightarrow}$ or $\overset{r_1 r_2 \dots r_n}{\Longrightarrow}$. The language generated by G is defined by $L(G) = \{w \in T^* | S \Rightarrow^* w\}$.

A very practical and typical way to represent a derivation in a grammar is a tree. Here, a derivation tree or a parse tree is defined as an ordered tree with the tree interior nodes as the left hand sides of grammar productions and the tree children nodes as the corresponding right hand side of that grammar where the tree root is the start symbol and the tree leaves are the terminals.

Formally, we can define a derivation tree as follows. Let $G = (N, T, S, P)$ be a context-free grammar and $S \Rightarrow^* w$ be a derivation in G. Then, a derivation tree of $S \Rightarrow^* w$ is an ordered and directed tree whose nodes are assigned with symbols of $N \cup T \cup \{\lambda\}$ in a manner like

any nonterminals of N are the interior nodes
the start symbol S is the root and
$A \rightarrow x_1 x_2 \dots x_n$ is a production of P if $x_1, x_2, \dots, x_n$ are the nodes children of nonterminal A which ordered from left to right.

Then, a derivation tree yield is the string over $N \cup T$ constructed by reading the leaves nodes starting from left to right.

Further, some regulated grammars that will be used in the following sections are defined here.

A matrix grammar is a quadruple $G = (N, T, S, M)$ where N, T and S are defined as for a context-free grammar and M is a set of matrices, which are sequences of context-free rules over $N \cup T$. The language generated by G is defined by $L(G) = \{w \in T^* | S \overset{\pi}{\Rightarrow} w \text{ and } \pi \in M^*\}$.

A tree controlled grammar is a quintuple $H = (N, T, P, S, R)$ where $G = (N, T, P, S)$ is a context-free grammar and $R \subseteq (N \cup T)^*$ is a regular set. The language $L(H)$ consists of all words w generated by the underlying grammar G such that there is a derivation tree t of w with respect to G, where the words of all levels (except the last one) are in R.

An additive valence grammar is a quintuple $G = (N, T, S, P, v)$ where N, T, S, P are defined as for context-free grammars and v is a mapping from P into the set $\mathbb{Z}$ of integers. The language generated by G consists of all strings $w \in T^*$ such there is a derivation $S \overset{r_1 r_2 \dots r_k}{\Longrightarrow} w$ such that $\sum_{i=1}^{k} v(r_i) = 0$.

The families of languages generated by matrix, tree controlled, additive valence grammars (with erasing rules) are denoted by $\mathbf{MAT}, \mathbf{TC}, a\mathbf{VAL}$ ($\mathbf{MAT}^{\lambda}, \mathbf{TC}^{\lambda}, a\mathbf{VAL}^{\lambda}$) respectively.

III. DEFINITIONS AND EXAMPLE

In this section, we present a conscientious definition of a tree valence controlled grammar, then we illustrate its features by an example.

Lemma 1: Let, for integers $v_1, v_2, \dots, v_n$, $\sum_{i=1}^{n} v_i = 0$. Then, there are decompositions of $v_i, 1 \leq i \leq n$, $v_i = \sum_{j=1}^{m} v_{ij}$ such that $v_{ij} \in \{-1, 0, 1\}$ for all $1 \leq i \leq n$, and $1 \leq j \leq m$, and $\sum_{i=1}^{n} v_{ij} = 0$ for all $1 \leq j \leq n$.

Proof : Let $v_1, v_2, \dots, v_n$ be integers where

$$v_1 + v_2 + \dots \dots + v_n = 0.$$

Then, every $v_i, 1 \leq i \leq n$ can be decomposed into a chain of valences v_{ij} consisting of either 0 and 1's for $v_i > 0$ or -1 and 0s for $v_i < 0$ in form $\begin{bmatrix} v_{i1} \\ v_{i2} \\ \vdots \\ v_{im} \end{bmatrix}$ where $\sum_{j=1}^{m} v_{ij} = v_i$.

In general, $v_1 + v_2 + \dots + v_n = 0$ can be decomposed into

$$\begin{array}{ccccccc} v_{11} & + & v_{21} & + & \cdots & + & v_{n1} \\ v_{12} & + & v_{22} & + & \cdots & + & v_{n2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{1m} & + & v_{2m} & + & \cdots & + & v_{nm} \end{array}$$

Furthermore, without considering the valence zero (0), the position of each valence v_{ij} that have the same sign (either -1 or 1) are placed in such a way that each of them are always at different level of height to each other. The size of height level is depending on the total number of one of the valence 1 or -1. For instance, if we have $v_1 = 3$ and $v_2 = 2$, then the size of height level will be 5 and their compositions will be

$$v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ and } v_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

It means, at every level, there will only have one valence 1 and one valence -1 with unbothered 0 (can have as many as we want). Then, the summation of all valences in each level is zero, i.e., $\sum_{j=1}^{m} \sum_{i=1}^{n} v_{ij} = 0$.

Generally, for any

$$v_1 + v_2 + \dots + v_n = 0$$

We can have

$$\begin{array}{ccccccccc} v_{11} & + & v_{21} & + & \cdots & + & v_{n1} & = & 0 \\ v_{12} & + & v_{22} & + & \cdots & + & v_{n2} & = & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{1m} & + & v_{2m} & + & \cdots & + & v_{nm} & = & 0 \end{array}$$

For example,

$$-3 + 2 + 1 + 2 + -2$$

$$\begin{array}{ccccccccc} -1 & + & 1 & + & 0 & + & 0 & + & 0 = 0 \\ -1 & + & 1 & + & 0 & + & 0 & + & 0 = 0 \\ -1 & + & 0 & + & 1 & + & 0 & + & 0 = 0 \\ 0 & + & 0 & + & 0 & + & 1 & + & -1 = 0 \\ 0 & + & 0 & + & 0 & + & 1 & + & -1 = 0 \end{array}$$

Definition 1 A tree valence controlled grammar is a quintuple $G = (N, T, S, P, V)$ where $G = (N, T, S, P)$ is a context-free grammar and V is a mapping from P into the set $\mathbb{Z}$ of integers.

Definition 2 The language $L(G)$ consists of all strings w generated by the underlying grammar G such that there is a derivation tree t of w with respect to G where the string w_i at every level $i \geq 1$ is obtained by production rules with zero total valence, i.e, if productions $r_{i,1}, r_{i,2}, \dots, r_{i,k_i} \in P, k_i \geq 1$, produces w_i, then $\sum_{j=1}^{k_i} V(r_{i,j}) = 0$.

The family of languages generated by tree valence controlled grammar (with erasing rules) is denoted by $\mathbf{TV}$ ($\mathbf{TV}^{\lambda}$).

Example 1 Let $G_1 = (\{A, B, C, S\}, \{a, b, c\}, S, P, V)$ be a tree valence controlled grammar where P consists of the following productions (for the sake of the simplicity, we assign the valence of a production in the brackets to the right of the production or over the arrow in the production)

$r_0 : S \to ABC[0]$,
$(S \overset{0}{\to} S_1,\ S_1 \overset{0}{\to} ABC)$,
$r_1 : A \to aA[2]$,
$(A \overset{1}{\to} A_1,\ A_1 \overset{1}{\to} aA)$,
$r_2 : B \to bB[-1]$,
$(B \overset{-1}{\to} B_1,\ B_1 \overset{0}{\to} bB)$,
$r_3 : C \to cC[-1]$,
$(C \overset{0}{\to} C_1,\ C_1 \overset{-1}{\to} cC)$,
$r_4 : A \to a[0]$,
$(A \overset{0}{\to} A_1,\ A_1 \overset{0}{\to} a)$,
$r_5 : B \to b[0]$,
$(B \overset{0}{\to} B_1,\ B_1 \overset{0}{\to} b)$,
$r_6 : C \to c[0]$,
$\left(C \overset{0}{\to} C_1,\ C_1 \overset{0}{\to} c\right)$.

We start with the only applicable production rule r_0 which yields ABC. Then we can either

terminate the derivation by applying productions $r_4 r_5 r_6$ to obtain a, b and c or

rewrite ABC to $aAbBcC$ by applying productions $r_1 r_2 r_3$.

Then, the derivation can be continued from A to C applying productions $r_1 r_2 r_3$ to produce a, b and c any number of times. To terminate the derivation, productions $r_4 r_5 r_6$ can be applied.

Then, we have the following derivation

Start:

$S \overset{r_0}{\Rightarrow} S_1 \Longrightarrow ABC.$

Iteratively generate:

$\overset{r_1}{\Rightarrow} A_1BC \Longrightarrow aABC \overset{r_2}{\Rightarrow} aAB_1C \Longrightarrow aAbBC$
$\overset{r_3}{\Rightarrow} aAbBC_1 \Longrightarrow aAbBcC$
$\overset{(r_1 r_2 r_3)^*}{\Longrightarrow} a^n A b^n B c^n C.$

Terminate:

$\overset{r_4}{\Rightarrow} a^n A_1 b^n B c^n C \Longrightarrow a^{n+1} b^n B c^n C$
$\overset{r_5}{\Rightarrow} a^{n+1} b^n B_1 c^n \Longrightarrow a^{n+1} b^{n+1} c^n C$
$\overset{r_6}{\Rightarrow} a^{n+1} b^{n+1} c^n C_1 \Longrightarrow a^{n+1} b^{n+1} c^{n+1}.$

Thus, G_1 generates the language

$$L(G_1) = \{a^n\, b^n c^n : n \geq 1\} \in \mathbf{CS} - \mathbf{CF}.$$

For instance, the string $a^3b^3c^3$ is obtained by the following derivation :

$S \overset{r_0}{\Rightarrow} S_1 \Longrightarrow ABC \in V \overset{r_1}{\Rightarrow} A_1BC$
$\Longrightarrow aABC \overset{r_2}{\Rightarrow} aAB_1C \Longrightarrow aAbBC \overset{r_3}{\Rightarrow} aAbBC_1$
$\Longrightarrow aAbBcC \in V \overset{r_1}{\Rightarrow} aA_1bBcC$
$\Longrightarrow aaAbBcC \overset{r_2}{\Rightarrow} aaAbB_1cC \Longrightarrow aaAbbBcC$
$\overset{r_3}{\Rightarrow} aaAbbBcC_1 \Longrightarrow aaAbbBccC \in V$
$\overset{r_4}{\Rightarrow} aaA_1bbBccC \Longrightarrow aaabbBccC$
$\overset{r_5}{\Rightarrow} aaabbB_1ccC \Longrightarrow aaabbbccC \overset{r_6}{\Rightarrow} aaabbbccC_1$
$\Longrightarrow aaabbbccC \in V = a^3b^3c^3.$

For more apparent, we present the derivations using tree diagram

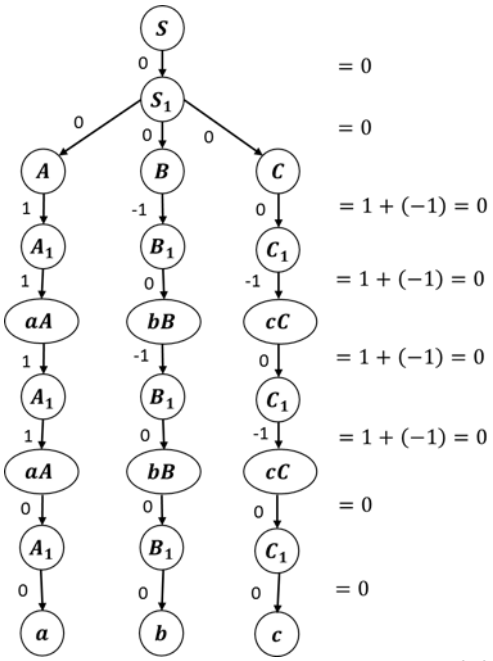

Figure 1 : Example of Tree Derivation for the String $a^3b^3c^3$.

IV. GENERATIVE POWERS

In this section, we manifest the results concerning the lower and upper bounds of tree valence controlled grammar defined above.

From the definitions, the next lemma follows immediately.

Lemma 2 $\mathbf{CF}^{[\lambda]} \subseteq \mathbf{TV}^{[\lambda]}$.

Proof : Let $G = (N, T, S, P)$ be a context-free grammar where $T = \{x_1, x_2, \dots x_n\}$. We construct the tree valence counter part of G as $G' = (N, T, S, P', V)$ where each $A \to w \in P$ is replaced with $A \to w[\omega], \omega \in \mathbb{Z}$ where ω then (by Lemma 1) is decomposed into $(a_1, a_2, a_3, \dots a_n), n \geq 1$ such $a_i : A_i \to w_i[a_i],\ a_i \in \{0, 1, -1\},\ 1 \leq i \leq n$ in P' and valence V is taken as $V(x_1, x_2, \dots x_n) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} = 0$. Thus, it is easy to see that $L(G) = L(G')$.

Example 2 Let $G_2 = (\{A, B, C, D, S\},\ \{a, b, c, d\},\ S,\ P,\ V)$ be a tree valence controlled context-free grammar where P consists of the following productions

$r_0 : S \to ABCD[0],$
$(S \overset{0}{\to} S_1,\ \ S_1 \overset{0}{\to} ABCD),$
$r_1 : A \to aA[1],$
$(A \overset{1}{\to} A_1,\ \ A_1 \overset{0}{\to} aA),$
$r_2 : B \to bB[1],$
$(B \overset{0}{\to} B_1,\ \ B_1 \overset{1}{\to} bB),$
$r_3 : C \to cC[-1],$
$(C \overset{-1}{\to} C_1,\ \ C_1 \overset{0}{\to} cC),$
$r_4 : D \to dD[-1],$
$(D \overset{0}{\to} D_1,\ \ D_1 \overset{-1}{\to} dD),$
$r_5 : A \to A[0],$
$(A \overset{0}{\to} A_1,\ \ A_1 \overset{0}{\to} A),$
$r_6 : B \to B[0],$
$(B \overset{0}{\to} B_1,\ \ B_1 \overset{0}{\to} B),$
$r_7 : C \to C[0],$
$(C \overset{0}{\to} C_1,\ \ C_1 \overset{0}{\to} C),$
$r_8 : D \to D[0],$
$(D \overset{0}{\to} D_1,\ \ D_1 \overset{0}{\to} D),$
$r_9 : A \to a[0],$
$(A \overset{0}{\to} A_1,\ \ A_1 \overset{0}{\to} a),$
$r_{10} : B \to b[0],$
$(B \overset{0}{\to} B_1,\ \ B_1 \overset{0}{\to} b),$
$r_{11} : C \to c[0],$
$(C \overset{0}{\to} C_1,\ \ C_1 \overset{0}{\to} c),$
$r_{12} : D \to d[0],$
$(D \overset{0}{\to} D_1,\ \ D_1 \overset{0}{\to} d),$

In general, we can have the derivation as follow :

Start:
$S \overset{r_0}{\Rightarrow} S_1 \Longrightarrow ABCD$

Generate:
$\overset{r_1}{\Rightarrow} A_1BCD \Longrightarrow aABCD \overset{r_6}{\Rightarrow} aAB_1CD \Longrightarrow aABCD$
$\overset{r_3}{\Rightarrow} aABC_1D \Longrightarrow aABcCD \overset{r_8}{\Rightarrow} aABcCD_1 \Longrightarrow aABcCD \in V$
$\overset{(r_1 r_6 r_3 r_8)^*}{\Longrightarrow} a^n ABc^n CD \in V$
$\overset{r_5}{\Rightarrow} a^n A_1 Bc^n CD \Longrightarrow a^n ABc^n CD \overset{r_2}{\Rightarrow} a^n AB_1 c^n CD$
$\Longrightarrow a^n AbBc^n CD \overset{r_7}{\Rightarrow} a^n AbBc^n C_1 D \Longrightarrow a^n AbBc^n CD$
$\overset{r_4}{\Rightarrow} a^n AbBc^n CD_1 \Longrightarrow a^n AbBc^n CdD \in V$
$\overset{(r_5 r_2 r_7 r_4)^*}{\Longrightarrow} a^n Ab^m Bc^n Cd^m D$

Terminate:
$\overset{r_9}{\Rightarrow} a^n A_1 b^m Bc^n Cd^m D \Longrightarrow a^{n+1} b^m Bc^n Cd^m D$
$\overset{r_{11}}{\Longrightarrow} a^{n+1} b^m Bc^n C_1 d^m D \Longrightarrow a^{n+1} b^m Bc^{n+1} d^m D$
$\overset{r_{10}}{\Longrightarrow} a^{n+1} b^m B_1 c^{n+1} d^m D \Longrightarrow a^{n+1} b^{m+1} c^{n+1} d^m D$
$\overset{r_{12}}{\Longrightarrow} a^{n+1} b^{m+1} c^{n+1} d^m D_1 \Longrightarrow a^{n+1} b^{m+1} c^{n+1} d^{m+1}.$

Plainly, this grammar generates a non-context free language : $L(G_1) = \{a^n\ b^m c^n d^m : n, m \geq 1\} \in \mathbf{CS} - \mathbf{CF}$.

From Lemmas 1 and 2 with Example 2, we obtain

Theorem 1 $\mathbf{CF}^{[\lambda]} \subsetneq \mathbf{TV}^{[\lambda]}$.

Next we show that tree valence controlled grammars can be simulated by tree controlled languages, i.e.,

Theorem 2 $\mathbf{TV}^{[\lambda]} \subseteq \mathbf{TC}^{[\lambda]}$.

Proof: Let $G = (N, T, S, P, V)$ be a tree valence controlled grammar. We construct an equivalent tree controlled grammar $G' = (N', T, S', P', R)$ where $N' = N \cup \{S', X, Y, Z\}$ with S', X, Y, Z are new non-terminals and $R \subseteq N^*$ is a regular set. We introduce the start TC

$$r_0 : S' \to S[Z] \qquad (1)$$

and define each production P_r of G'. For $r = A \to w[\omega] \in P$,

$$P_r : (A \to w, [\ Z \to X^{\sum_{j=1}^{m} v_{ij},\ v_{ij} > 0} Z\ ]) \text{ if } \omega = 1 \qquad (2)$$

and

$$P_r : (A \to w, [\ Z \to Y^{\sum_{j=1}^{m} v_{ij},\ v_{ij} < 0} Z \text{ if } \omega = -1 \qquad (3)$$

We control the use of a production at every level by

$$R = \{Z, X^{\sum_{j=1}^{m} v_{ij}} Y^{\sum_{j=1}^{m} v_{ij}} Z\} \qquad (4)$$

We also consider the erasing production

$$r_{\lambda,Z} : (Z \to \lambda) \quad (5)$$

and

$$r_{\lambda} : (X \to \lambda)\ and\ (Y \to \lambda) \quad (6)$$

The TC grammar consists of the productions defined in (1) – (4) above. Further, we show that $L(G) = L(G')$.

First, we show $L(G) \subseteq L(G')$. Let

$D: S \overset{r_0}{\Rightarrow} w_1[\omega_1] \overset{r_1}{\Rightarrow} w_2[\omega_2] \Rightarrow \ldots \ldots \overset{r_t}{\Rightarrow} w_t[\omega_t], w_t \in T^*$

be a successful derivation, where the sum $\sum_{i=1}^{t} \omega_i$ is represented in the following form according to Lemma 1:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij} = 0.$$

We construct a derivation D' in G' simulating D. The derivation D' starts with (1) and for each r_i in D, we choose P_{r_i} in D' with a derivation of tree TC of w with respect to G' where the words of all levels except the last one are belonging to R (meet condition (4)). i.e.,

$S' \overset{r_0}{\Rightarrow} SZ \overset{r_1}{\Rightarrow} w_1 X^{\sum_{j=1}^{m} v_{ij},\ v_{ij}>0} Y^{\sum_{j=1}^{m} v_{ij},\ v_{ij}<0} Z$
$\overset{r_2}{\Rightarrow} w_2 X^{\sum_{i=1}^{2}\sum_{j=1}^{m} v_{ij},\ v_{ij}>0} Y^{\sum_{i=1}^{2}\sum_{j=1}^{m} v_{ij},\ v_{ij}<0} Z$
$\overset{r_n}{\Rightarrow} w_n X^{\sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij},\ v_{ij}>0} Y^{\sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij},\ v_{ij}<0} Z \in R$
$\Rightarrow \ldots \ldots \overset{r_t}{\Rightarrow} w_t X^A Y^B Z$ where
$A = \sum_{S=1}^{t}(\sum_{j=1}^{m} v_{ij},\ v_{ij} > 0\ )$,
$B = \sum_{S=1}^{t}(\sum_{j=1}^{m} v_{ij},\ v_{ij} < 0\ )$.

Afterwards, we apply the erasing matrices (5) and (6) until Z, Xs and Ys are completely removed where

$$D' : S' \overset{*}{\Rightarrow} w_t X^A Y^B Z \overset{*}{\Rightarrow} w_t. \quad (7)$$

Derivation (7) with $A - B = 0$ is possible since at every level of tree, the derivation of tree are already in R.

From the other hand, we show that $L(G') \subseteq L(G, V, =)$.

We consider a successful derivation D' in G'. Any derivation in G' starts with applying $l_0 : r_0$, then any production from (2) – (3) can be applied with satisfying the production (4). Yet, as soon as production (5) is applied, matrices (2) further cannot be applied. Without loss of generality, we can assume that

$$D' : S' \overset{r_0}{\Rightarrow} SZ \overset{r_1 r_2 \ldots r_t}{\Longrightarrow} w_t X^A Y^B Z \overset{r_{\lambda,Z}}{\Longrightarrow} w X^A Y^B \overset{r_{\lambda}}{\Rightarrow} w_t$$

where
$A = \sum_{S=1}^{t}(\sum_{j=1}^{m} v_{ij},\ v_{ij} > 0\ )$ and
$B = \sum_{S=1}^{t}(\sum_{j=1}^{m} v_{ij},\ v_{ij} < 0\ )$.

Since a derivation of tree belong to R at every level except the last one, r_{λ} production erases all Xs and Ys with $A - B = 0$.

From the other hand,

$$A - B = \sum_{S=1}^{t}\left(\sum_{j=1}^{m} v_{ij},\ v_{ij} > 0\ \right) - \sum_{S=1}^{t}\left(\sum_{j=1}^{m} v_{ij},\ v_{ij} < 0\ \right) = 0.$$

Then, the corresponding derivation in G is $D : S \overset{r_0 r_1 \ldots r_t}{\Longrightarrow} w$.

Next theorem establishes a better upper-bound for the family of tree valence languages.

Theorem 3 $\mathbf{TV}^{[\lambda]} \subseteq \mathbf{MAT}^{[\lambda]}$.

Proof: Let G be a tree valence controlled grammar where $G = (N, T, S, P, V)$ and $L' = L(G)$. We construct an equivalent matrix grammar $G' = (N', T, S', M')$ where $N' = N \cup \{S', R, Z\}$ where S', R, Z are new non-terminals. We introduce the start matrix

$$m_0 : (S' \to SZ) \quad (1)$$

and define the matrix m_r for each production

$r = A \to w[\omega] \in P$, the matrix

$$m_r : (A \to w, [\ Z \to R^{\sum \omega(a_i)} Z\ ]) \text{ for } a_{ij} = 1 \quad (2)$$

We also consider the erasing matrices

$$m_{\lambda} : (R \to \lambda) \text{ for } a_{ij} = -1, \quad (3)$$
$$m_{\lambda,Z} : (Z \to \lambda), \quad (4)$$

The matrix set M' consists of the matrices (1) – (4) defined above. Further, we show that $L(G) = L(G')$. First we present $L(G) \subseteq L(G')$.

Let $D: S \overset{r_0}{\Rightarrow} w_1[\omega_1] \overset{r_1}{\Rightarrow} w_2[\omega_2] \Rightarrow \ldots \ldots \overset{r_t}{\Rightarrow} w_t[\omega_t], w_t \in T^*$ be a successful derivation, i.e., $\sum_{i=1}^{n} a_{ij} = 0$ for all $1 \le j \le m$.

We construct a derivation D' in G' simulating D. D' starts with matrix (1) and for each r_i in D, we choose m_{r_i} in D', i.e.,

$S' \overset{m_0}{\Rightarrow} SZ \overset{m_{r_1}}{\Longrightarrow} w_1 R^{\sum \omega_1(a_i)} Z \overset{m_{r_2}}{\Longrightarrow} w_2 R^{\sum(\omega_1(a_i)+\omega_2(a_i))} Z$
$\Rightarrow \ldots \ldots \overset{m_{r_t}}{\Longrightarrow} w_t R^A Z$ where $A = \sum_{S=1}^{t}(\sum \omega_S(a_i))$.

Afterwards, we apply the erasing matrices (3) and (4) until Z and Rs are completely removed where

$$D' : S' \overset{*}{\Rightarrow} w_t R^A Z \overset{*}{\Rightarrow} w_t.$$

Derivation (5) is possible since
$A = \sum_{i=1}^{n} a_{ij} = 0$ for all $1 \le j \le m$.

From the other hand, we show that $L(G') \subseteq L(G)$.

We consider a successful derivation D' in G'. Any derivation in G' starts with applying m_0, then any matrix

from (2) – (4) can be applied. Yet, as soon as matrix (4) is applied, matrices (2) further cannot be applied. Without loss of generality, we can assume that

$$D' : S' \overset{m_0}{\Rightarrow} SZ \overset{m_{r_1} m_{r_2} \dots m_{r_t}}{\Longrightarrow} w_t R^A Z \overset{m_{\lambda,Z}}{\Longrightarrow} w_t R^A \overset{m_\lambda}{\Rightarrow} w_t$$

where $A = \sum_{S=1}^{t} (\sum \omega_S(a_i))$.

Since m_λ matrix erases all Rs, $A = 0$.

Then, the corresponding derivation in G is $D : S \overset{r_1 r_2 \dots r_t}{\Longrightarrow} w$.

Next, we give an example to illustrate the idea of construction the matrix grammar for a tree valence controlled grammar.

Example 4 Consider the language $L(G_4) = \{a^n b^n c^n d^n e^n f^n : n \geq 0\} \in \mathbf{CS} \cap \mathbf{TV} - \mathbf{CF}$ generated by tree valence controlled grammar $G_4 = (\{A, B, C\ S\},\ \{a, b, c,\ d, e, f\},\ S,\ P, V\ )$ with production

$r_0 : S \to ABC[0]$,
$(S \overset{0}{\to} S_1,\ S_1 \overset{0}{\to} ABC)$,
$r_1 : A \to aAb[2]$,
$(A \overset{1}{\to} A_1, A_1 \overset{1}{\to} aAb)$,
$r_2 : B \to cBd[-1]$,
$(B \overset{-1}{\to} B_1,\ B_1 \overset{0}{\to} cBd)$,
$r_3 : C \to eCf[-1]$,
$(C \overset{0}{\to} C_1,\ C_1 \overset{-1}{\to} eCf)$,
$r_4 : A \to \lambda[0]$,
$(A \overset{0}{\to} A_1, A_1 \overset{0}{\to} \lambda)$,
$r_5 : B \to \lambda[0]$,
$(B \overset{0}{\to} B_1, B_1 \overset{0}{\to} \lambda)$,
$r_6 : C \to \lambda[0]$,
$(C \overset{0}{\to} C_1,\ C_1 \overset{0}{\to} \lambda)$.

We construct the matrix grammar $G_4' = (\{A, B, C, S\}, \{a, b, c, d, e, f\},\ S,\ P,\ M\ )$ simulating G_4 with production such

$m_0 : (S' \to SZ)$,
$m_1 : (S \to ABC[Z \to Z])$,
$m_2 : (A \to aAb[Z \to R_2 R_3 Z])$,
$m_3 : (B \to cBd[R_2 \to \lambda])$,
$m_4 : (C \to eCf[R_3 \to \lambda])$,
$m_5 : (A \to \lambda)$,
$m_6 : (B \to \lambda)$,
$m_7 : (C \to \lambda)$,
$m_8 : (Z \to \lambda)$.

Now, we demonstrate the derivation of those two grammars using a string $a^2 b^2 c^2 d^2 e^2 f^2$.

By tree valence controlled grammar

$S \overset{r_0}{\Rightarrow} S_1 \Longrightarrow ABC \in V$
$\overset{r_1}{\Rightarrow} A_1 BC \Longrightarrow aAbBC \overset{r_2}{\Rightarrow} aAbB_1 C$
$\Longrightarrow aAbcBdC \overset{r_3}{\Rightarrow} aAbcBdC_1$
$\Longrightarrow aAbcBdeCf \in V$
$\overset{r_1}{\Rightarrow} aA_1 bcBdeCf \Longrightarrow aaAbbcBdeCf$
$\overset{r_2}{\Rightarrow} aaAbbcB_1 deCf \Longrightarrow aaAbbccBddeCf$
$\overset{r_3}{\Rightarrow} aaAbbccBddeC_1 f$
$\Longrightarrow aaAbbccBddeeCff \in V$
$\overset{r_4}{\Rightarrow} aA_1 bcBdeCf \Longrightarrow aabbcBdeCf$
$\overset{r_5}{\Rightarrow} aabbcB_1 deCf \Longrightarrow aabbccddeeCff$
$\overset{r_6}{\Rightarrow} aaAbbccddeC_1 f \Longrightarrow aabbccddeeff \in V$
$= a^2 b^2 c^2 d^2 e^2 f^2$.

By matrix grammar

$S' \overset{m_0}{\Rightarrow} SZ \overset{m_1}{\Rightarrow} ABCZ \overset{m_2}{\Rightarrow} aAbBCR_2 R_3 Z$
$\overset{m_3}{\Rightarrow} aAbcBdCR_3 Z \overset{m_4}{\Rightarrow} aAbcBdeCfZ$
$\overset{m_2}{\Rightarrow} aaAbbcBdeCfR_2 R_3 Z$
$\overset{m_3}{\Rightarrow} aaAbbccBddeCfR_3 Z$
$\overset{m_4}{\Rightarrow} aaAbbccBddeeCffZ$
$\overset{m_5}{\Rightarrow} aabbccBddeeCffZ$
$\overset{m_6}{\Rightarrow} aabbccddeeCffZ$
$\overset{m_7}{\Rightarrow} aabbccddeeffZ \overset{m_8}{\Rightarrow} aabbccddeff$
$= a^2 b^2 c^2 d^2 e^2 f^2$.

The language $L(G) = \{a^n b^n c^n \mid n \geq 1\}^2$ has been proven cannot be generated by an additive valence grammar by Dassow and Paun (1989) in example 2.1.7. Nevertheless, this language can be generated by tree valence controlled grammar as in Example 5.

Example 5

$L(G_5) = \{a^n b^n c^n \mid n \geq 1\}^2 \in \mathbf{TV} - a\mathbf{VAL}$.

The grammar G_5 for $L(G_5)$:

$r_0 : S \to ABCD[0]$,
$(S \overset{0}{\to} S_1,\ S_1 \overset{0}{\to} ABCD)$,
$r_1 : A \to aAb[1]$,
$(A \overset{1}{\to} A_1,\ A_1 \overset{0}{\to} aAb)$,
$r_2 : B \to cB[-1]$,
$(B \overset{-1}{\to} B_1,\ B_1 \overset{0}{\to} cB)$,
$r_3 : C \to aCb[1]$,
$(C \overset{0}{\to} C_1,\ C_1 \overset{1}{\to} aCb)$,
$r_4 : D \to cD[-1]$,
$(D \overset{0}{\to} D_1,\ D_1 \overset{-1}{\to} cD)$,
$r_5 : A \to ab[0]$,
$(A \overset{0}{\to} A_1,\ A_1 \overset{0}{\to} ab)$,
$r_6 : B \to c[0]$,
$(B \overset{0}{\to} B_1,\ B_1 \overset{0}{\to} c)$,
$r_7 : C \to ab[0]$,
$(C \overset{0}{\to} C_1,\ C_1 \overset{0}{\to} ab)$,
$r_8 : D \to c[0]$,
$(D \overset{0}{\to} D_1,\ D_1 \overset{0}{\to} c)$.

Here, obviously we can have the derivation such

Start:

$S \overset{r_0}{\Rightarrow} S_1 \Longrightarrow ABCD$

Generate:

$\overset{r_1}{\Rightarrow} A_1BCD \Longrightarrow aAbBCD \overset{r_2}{\Rightarrow} aAbB_1CD \Longrightarrow aAbcBCD \in V$

$\overset{(r_1r_2)^*}{\Longrightarrow} a^nAb^nc^nBCD$

$\overset{r_3}{\Rightarrow} a^nAb^nc^nB_1CD \Longrightarrow a^nAb^nc^nBaCbD$

$\overset{r_4}{\Rightarrow} a^nAb^nc^nBaCbD_1 \Longrightarrow a^nAb^nc^nBaCbcD \in V$

$\overset{(r_3r_4)^*}{\Longrightarrow} a^nAb^nc^nBa^mCb^mc^mD$

Terminate:

$\overset{r_5}{\Rightarrow} a^nA_1b^nc^nBa^mCb^mc^m$

$\Longrightarrow a^{n+1}b^{n+1}c^nBa^mCb^mc^mD$

$\overset{r_6}{\Rightarrow} a^{n+1}b^{n+1}c^nB_1a^mCb^mc^mD$

$\Longrightarrow a^{n+1}b^{n+1}c^{n+1}a^mCb^mc^mD$

$\overset{r_7}{\Rightarrow} a^{n+1}b^{n+1}c^{n+1}a^mC_1b^mc^mD$

$\Longrightarrow a^{n+1}b^{n+1}c^{n+1}a^{m+1}b^{m+1}c^mD$

$\overset{r_8}{\Rightarrow} a^{n+1}b^{n+1}c^{n+1}a^{m+1}b^{m+1}c^mD_1$

$\Longrightarrow a^{n+1}b^{n+1}c^{n+1}a^{m+1}b^{m+1}c^{m+1}.$

Thus, G_5 generates the language

$L(G_5,) = \{ a^nb^nc^n \mid n \geq 1 \}^2.$

From Example 5, it follows that

Theorem 4 $\mathbf{TV} - a\mathbf{VAL} \neq \emptyset$.

V. CONCLUSION

In a nut shell, we have introduced a new variant of controlled grammars called tree valence controlled grammar in which its basic notion is based on the existing of well-known and well-developed controlled grammar named tree controlled grammar. Here, we have found that tree valence controlled grammars are more powerful than context free grammars as well as than additive valence grammar. Moreover, they also generate matrix languages.

ACKNOWLEDGMENT

This research has been supported by the grants RIGS16-368-0532 and FRGS13-074-0315 of Ministry of Education, Malaysia through International Islamic University Malaysia.

REFERENCES

[1] K. Culik II and H.A. Maurer. "Tree controlled grammars," *Computing*, vol. 19, pp.129–39, 1977.

[2] G. Paun. "On the generative capacity of tree controlled grammars," *Computing*, vol. 22, pp. 213-220, 1979.

[3] J. Dassow and B. Truthe, "Subregularly tree controlled grammars and languages," in *Proc E. Csuhaj-Varju, Z.Esik (Eds): Automata and Formal Languages (AFL'08)*, 2008, pp. 158-169.

[4] J. Dassow and B. Truthe, "On two hierarchies of subregularly tree controlled languages," in *Proc C. Campeanu, G. Pighizzini (eds): Descriptional Complexity of Formal Systems*, 2008, pp. 145–56.

[5] R. Stiebe. "On the complexity of the control language in tree controlled grammars," in *Proc J. Dassow, B. Truthe (Eds): Colloquium on the Occasion of the 50th Birthday of Victor Mitrana*, 2008, pp. 29–36.

[6] S. Turaev, J. Dassow and M.H. Selamat , "Nonterminal complexity of tree controlled grammars," *Theoritical Computer Science*, vol. 412, pp. 5789–5795, 2011.

[7] S. Turaev, J. Dassow and M.H. Selamat, "Language classes generated by tree controlled grammars with bounded nonterminal complexity," *Descriptional Complexity of Formal System*, vol. 6808, pp. 289–300, 2011.

[8] S. Turaev, J. Dassow , F. Manea and M.H. Selamat, "Language classes generated by tree controlled grammars with bounded nonterminal complexity," *Theoritical Computer Science*, vol. 449, pp. 134–44, 2012.

[9] G. Vaszil, " On the nonterminal complexity of tree controlled grammars," in *Bordihn H., Kutrib M., Truthe B. (eds) Language Live*, ser. Lecture Notes in Computer Science. Springer Verlag:New York, vol. 7300, pp. 265–272. 2012.

[10] J. Koutny and A. Meduna, "Tree controlled grammars with restrictions placed upon cuts and paths," *Kybernetika*, 48(1), pp. 165–75, 2012.

[11] A. Meduna and P. Zemek, *Regulated Grammars and Automata*, New York: Springer-Verlag Heidelberg, 2014.

[12] J. Dassow and G. Paun. *Regulated Rewriting in Formal Language Theory*, New York: Springer-Verlag Berlin Heidelberg, 1989.

[13] A. Salomaa, *Formal Languages*, New York: New York Academic Press, 1973.

[14] G. Bel-Enguix, M. Jimenez-Lopez and C. Martin-Vide, *New Developments in Formal Languages and Applications*, New York: Springer Berlin Heidelberg, 2008.

[15] M. Sipser, *Introduction to the Theory of Computation*, 3rd Ed., United States of America: Cengage Learning, 2013.