Structurally and Arithmetically Controlled Grammars

S. Ashaari[1], S. Turaev[1], A. Okhunov[2]
[1]Department of Computer Science, Kulliyyah of Information and Communication
[2]Department of Science in Engineering, Kulliyyah of Engineering
International Islamic University Malaysia, Kuala Lumpur, Malaysia

Abstract— Over the quarter century, it is gratifying to note that the significance of regulated or controlled grammars (i.e. grammars with regulated rewriting) has been recognized by many parties where it has been used widely in a great variety of scientific disciplines ranging from Linguistics through DNA Computing up to the Informatics and recently come to Big Data Analytics. Therefore, literally we can find hundreds of studies of well-known of various types of controlled grammars and their investigation have amount to a thrilling trend within formal language theory. Given the extensive literature on issues related to controlled grammars, this research focused on arithmetically controlled grammars and tree controlled grammars, which are practically important. Thereby, in this paper, we briefly recapitulate the background of formal language theory and highlight the key results of multiset grammars, valence grammars and tree controlled grammars. In this paper, a new controlled grammar that can be generated using both control mechanisms together is proposed for future research.

Keywords— Multisets, Valences, Weights, Trees, Controlled Grammars.

I. INTRODUCTION

The study of formal language theory which started in the middle of the 20th century and primarily originated from mathematics [1] where later on has emerged in disciplines of linguistics, computer science and biology has been widely recognized as the stem of theoretical computer science in the sense that all human problems can be considered as symbol manipulation, and as structures formulated by symbols. It was born in three phases: In 1956, Chomsky [2] claimed a new way of looking at natural languages syntax by proposing a generative device hierarchy. Subsequently, Ginsburg and Rice [3] demonstrated that the hierarchy of Chomsky is applicable not only for natural languages but it is also able to cope with the problems of semantic and syntactic of programming languages too in 1961. Because of that, the theory of formal grammars and languages has become nearly identical to the programming languages theory. Later, Salomaa [4] presented a chapter of theoretical computer science that contains a complete and accurate mathematical clarification of the theory in 1973. Since then, the theory of formal language has been an active and growing research area with broad applications in fields such as pattern recognition, cryptography, compiler, computer networks, artificial life, molecular computing, image enhancement or compression and many more until today [5, 6].

In principle, the theory of formal language consists of two basic approaches, which are grammatical and automata approach. Grammatical approach, which is better known as grammars, are language generation models that define their language strings so their process of rewriting will generate them starting from a special start symbol. On the other hand, automata approach, the language recognition models define their language strings by a process of recognition that starts from the initial state and ends in a final state [5]. In this paper, we focus on the grammar formalism.

A grammar is naturally a set of rules (or productions) used to construct a language over a certain alphabet Σ, where different language types can be developed depending on the way of rewriting rules. It also contains the other three principal constructs, which are sets of nonterminals, terminals and productions. A nonterminal is written in upper case (eg : S, A) and it represents the grammar's state, and indicate which productions can be used afterwards. A terminal is written in lower case (eg: a, b) that delineates a value generated by the grammar in the final string and a production in form of $A \rightarrow w$ is a combination series of terminals and nonterminals where the left hand side must have at least one nonterminal symbol and it works in such a value from left hand side is turning into the right hand side value. In addition, in term of derivation to generate a string of a language, it will start with a string that consists of a single start symbol S where

then any rule can be applied in any order to replace the nonterminal symbol in right hand side as long as it is a substring of the designated left hand side. Then, it is said to be a complete derivation, when there exists only terminal symbols in the current string. The set of all generated strings will defined the language type. In fact, any certain sequence of rules on the start symbol will produce a distinct string in the language and a grammar is called ambiguous if it can generate the same single string in multiple ways [6-8].

Indeed, a containment hierarchy of grammars classes that well known and received most attention in formal models field is referred as Chomsky hierarchy. In Chomsky hierarchy, grammars are categorized into four main classes such as regular which decided by finite automaton, context-free which recognized by pushdown automaton, context-sensitive which acknowledged by linear bounded automaton and recursively enumerable (aka unrestricted) which accepted by Turing machine according to the order of increasing complexity where a language of higher order contains a subset of all languages of lower complexity. In other words, there exist context-free languages which are not regular, context-sensitive languages which are not context-free and as well as unrestricted languages which are not context-sensitive [7, 8].

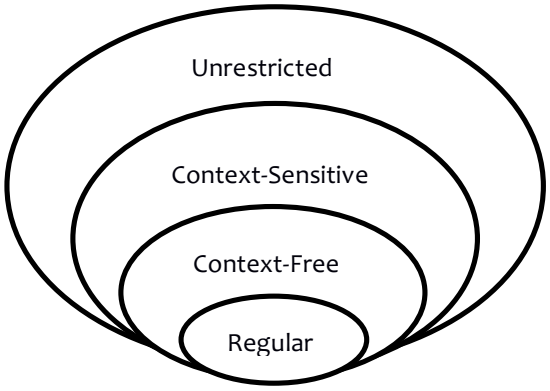

Fig. 1: Set Inclusion of Grammars Described by Chomsky Hierarchy

Then, in the Chomsky hierarchy, the context-free grammars which in the first place used to study human languages are the most flourishing and favouring class of grammars in the evolution of language models due to their beauty in term of simplicity and intuitively captivating formalism. They are a natural formalism that deputizes the constituency language behaviour where a constituent is clarified as a group of sequential of strings operating as a unit. Informally, a context-free grammar is a set of rules that grant one to substitute a variable by a string of terminals and variables where each string in the language own a derivation tree with leftmost derivation. In fact they have a broad applicability and the same time they have a great mathematical appeal [6, 9].

However, it is well known that the world is not "context-free" where there are many circumstances that caused the appearing of non-context-free languages which have the basic features like reduplication (e.g.: $\{ww|w \in T^*\}$) , multiple agreement (e.g: $\{a^n b^n c^n d^n | n \geq 1\}$) and crossed agreements (e.g: $\{a^n b^m c^n d^m | n, m \geq 1\}$) [10]. This situation has called for more powerful grammars with similar properties of context-free grammars known as grammars with regulated rewriting.

The first type of grammars with regulated rewriting, called matrix grammars, (i.e. controlled grammars or regulated grammars) was introduced by Abraham in 1965 [11]. These grammars use the same rules of grammars as in Chomsky hierarchy but have accompanied by certain additional mechanisms so that the application of rules can be restricted in order to avoid particular derivations. Since then, a lot of variants of grammars with regulated rewriting have been investigated and studied in formal language field where mostly are based on context-free grammar with the aim to increase their computational power so that they can cover more aspect of real application problems.

Despite their diversity, all of the introduced regulated grammars can be classified into several types depending on their common characteristics like (1) control by prescribed sequences such as matrix grammars [12-17], regularly controlled grammars [18], vector grammars [19], different variants of Petri net controlled grammars [20-26] and Parikh vector controlled grammars [27], (2) control by context conditions such as conditional grammars and ordered grammars [28], random context grammars [29], tree controlled grammars [30-37], semi-conditional grammars [38] and string-regulated graph grammars [39], (3) control by computed sequences such as programmed grammars [40] and valence grammars [41-47], (4) control by memory such as indexed grammars [48], (5) control by partial parallelism such as scattered context grammars [49], Russian parallel grammars [50], Indian parallel grammars [51] and global indexed grammars [52] and many other regulated grammars [5, 53]. Among all established controlled grammars, we interested on reviewing the multiset grammars, valence grammars and tree controlled grammars since they are simple but powerful grammars.

The rest of section in this paper is structured as following: In Section 2, we recall some well-known basic notations, terminologies and concept related to formal languages theory, multiset, tree and so on which will be used throughout this paper. In Sections 3, 4 and 5, we recall the original definition of multiset grammars, valence grammars and tree controlled grammars with their well-established computational power and closure properties as well as the relevant overview of research done in previous until current one which related to them. In Section 6, we

give a brief summarization of all materials discussed before and present some suggestions for future work.

II. PRELIMINARIES

In this section, we shortly recall the necessary basic notations, notions and definitions related to formal languages theory, multiset, and a derivation tree which will be used in the following section. For more detailed information, the reader is referred to [6-9, 12].

A. General Notation

Throughout the paper, we use the following notations. In the set theory area, we have the symbols $\in$ and $\notin$ to represent the set membership and negation of set membership of an element to a set; $\subseteq$ to signify the inclusions which is not necessarily proper and $\subset$ to mark for the strict inclusion; $|A|$ to portray the cardinality of a set A which is the number of elements in the set A and 2^A to depict the power set of a set A; $\emptyset$ to denote the empty set which implies that there is no elements in the set. Then in the set of numbers field, we have the set of integer, natural, real and rational number which are denoted by $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{R}$ and $\mathbb{Q}$. Next, we have an alphabet which is a finite and nonempty set of elements known as symbols or letters in which denoted by Σ and a string (sometimes referred as word) over the alphabet Σ, which is a finite sequence of symbols (concatenation of symbols) from Σ. Then, the formed 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 $\Sigma^*$and the set of all non-erasing strings is denoted by Σ^+, i.e., $\Sigma^+ = \Sigma^* - \{\lambda\}$. A language L is a subset of Σ^* and $L \subseteq \Sigma^*$ is termed λ -free if $\lambda \notin L$.

B. Grammars

A phrase structure grammar is a quadruple $G = (N, T, S, P)$, where N is an alphabet of non-terminals, T is an alphabet of terminals, S is the start symbol where $S \in N$, P is a finite set of production of the form $A \to w$ where $A \in (N \cup T)^+$, $w \in (N \cup T)^*$ and with the condition that A must contains at least one non-terminal symbol. If a production is in form of $A \to \lambda$ then it is called an erasing rule.

For a grammar $G = (N, T, S, P)$, a direct derivation relation over $(N \cup T)^*$ which denoted by $\Rightarrow$ and defined as $u \Rightarrow v$ provided if and only if there is a rule $A \to w \in P$ such that $u = x_1 A x_2$ and $v = x_1 w x_2$ for $x_1, x_2 \in (N \cup T)^*$. Since $\Rightarrow$ is a relation, then the nth power of $\Rightarrow$ is $\Rightarrow^n$ for $n \geq 0$ where $\Rightarrow^+$ is known as transitive closure and $\Rightarrow^*$ as reflexive-transitive closure.

A derivation that use the sequence of rules $m = r_0 r_1 \dots r_n$, $r_i \in P, 1 \leq i \leq n$, is denoted by $\overset{m}{\Rightarrow}$ or $\overset{r_0 r_1 \dots r_n}{\Longrightarrow}$. Then, a string $w \in (N \cup T)^*$ is a sentential form if $S \Rightarrow^* w$. If $w \in T^*$, then w is called a sentence or a terminal string and $S \Rightarrow^* w$ is said to be a successful derivation.

The language generated by G denoted by $L(G)$ is defined as $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$.

Two grammars G_1 and G_2 are called to be equivalent if and only if they generate the same language such that $L(G_1) = L(G_2)$.

The Chomsky hierarchy classifies all grammars into four basic categories according to their complexity of production rules, i.e., a grammar $G = (N, T, S, P)$ is called

unrestricted or recursively enumerable grammar (type-0) if its productions are in the form of $A \to w$ where $A \in (N \cup T)^+$, $w \in (N \cup T)^*$ and A contains at least one non-terminal symbol.

context sensitive grammar (type-1) if its productions are in the form of $A \to w$ where $|A| \leq |w|$, $A \in (N \cup T)^* N^+ (N \cup T)^*$ and $w \in (N \cup T)^+$.

context free grammar (type-2) if its productions are in the form of $A \to w$ where $A \in N$ and $w \in (N \cup T)^*$.

linear grammar if its productions are in the form of $A \to w$ where $A \in N$ and $w \in T^* \cup T^* N T^*$.

regular grammar (type-3) if its productions are in the form of $A \to w$ where $w \in T^* \cup T^* N$ and $A \in N$.

The families of languages generated by arbitrary, unrestricted, context sensitive, context free, regular, linear and finite grammars are denoted by **RE**, **CS**, **CF**, **REG**, **LIN** and **FIN** , respectively. For these language families, Chomsky hierarchy holds:

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

A parallel rewriting system called an $ETOL$ system is defined as a construct $G = (N \cup T, T, P_1, P_2, \dots, P_m, S)$ where N, T, S are defined as normal context-free grammars with P_i as a finite subset of $(N \cup T) \times (N \cup T)^*$ such that for each $\alpha \in N \cup T$, there is at least one pair (α, β) take place in P_i. The pairs in P_i are known as productions and written as $\alpha \to \beta$. Then, for an arbitrary word $w = a_1 a_2 \dots a_n$, $a_i \in N \cup T$ with productions $a_1 \to \beta_1, a_2 \to \beta_2, \dots, a_n \to \beta_n$ of the equal set P_k, we write $a_1 a_2 \dots a_n \Rightarrow \beta_1 \beta_2 \dots \beta_n$ and denote its reflexive-transitive closure of $\Rightarrow$ by $\Rightarrow^*$. The language generated by this G is defined same as phrase structure grammar such $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$.

Then, an $ETOL$ system that consists of only a single set of productions is known an EOL system. The languages generated by both systems are noted as $ETOL$ languages and EOL languages respectively.

Henceforth, a retrospect of some definition of regulated grammars that will be used in the following sections will be defined here and forward.

A matrix grammar is a quadruple $G_m = (N, T, S, M)$ where N, T and S are defined exactly as for a normal context-free grammar and M is a set of matrices, sequences of context-free rules over $N \cup T$. The language generated by G_m is defined by

$$L(G) = \{w \in T^* \mid S \overset{\pi}{\Rightarrow} w \text{ and } \pi \in M^*\}.$$

An unordered vector grammar is a quadruple $G = (N, T, S, M)$ where N, T, S and M are defined as for matrix grammars. Then, the language $L(G)$ generated by a derivation such

$$S \Rightarrow_{p_1} w_1 \Rightarrow_{p_2} w_2 \Rightarrow_{p_3} \dots \Rightarrow_n w$$

where $p_1 p_2 \dots p_n$ is a permutation of a string in M^*.

The families of languages generated by matrix, unordered vector grammars (with erasing rules) are denoted by $\mathbf{MAT}, \mathbf{uV}$ $(\mathbf{MAT}^{\lambda}, \mathbf{uV}^{\boldsymbol{\lambda}})$, respectively.

C. Derivation Tree

A derivation tree of $S \Rightarrow^* w$ is an ordered and directed tree whose its 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.

III. MULTISET GRAMMARS

Multiset is defined in [54] as a collection of unordered objects (known as elements) that are allowed to have repeated occurrences of identical elements and its number of times of occurrences is called multiplicity. It is important to consider the term multiset since there exist circumstances such there are repeated hydrogen and oxygen atoms in a sulfuric acid molecule (H_2SO_4), repeated roots of polynomial equations, repeated observations in statistical samples and so on where those repeated elements are needed to be counted in order to attain their definiteness and adequacy. On account to its aptness [54, 55], multiset has been used interchangeably with a variety of term that carrying a synonymy meaning with it even in different contexts like "heap", "bag", "occurrence set", "fireset", "weighted set", "sample", "bunch" and "list".

The notion of relating multiset rewriting with Chomsky grammars was initiated by Kudlek, Martin and Paun [56] in 2001 with the name of "multiset grammars" where they considered the applicability of rules as multiset in restricting the use of productions of grammars. The definition of multiset grammar with its computational powers and closure properties are demonstrated in the following definition and theorems.

Definition 1 [56] A multiset grammar is a quadruple $G = (N, T, S, P)$ where N and T are alphabets of nonterminals and terminals, S is the starting multiset and P is a finite set of multiset rewriting rules in form $\alpha \rightarrow \beta$ with α, β are multisets over $N \cup T$ and $\sum_{A \in N} \alpha(A) \geq 1$. For to multisets x_1, x_2 over $N \cup T$, we write $x_1 \Rightarrow x_2$ if there is a multiset rule $r : \alpha \rightarrow \beta \in$ P such that $\alpha \sqsubseteq x_1$ and $x_2 = x_1 - \alpha + \beta$. Then, the language generated by multiset grammar is defined by $L(G) = \{w \in T^{\oplus} \mid S \Rightarrow^* w\}$.

The families of language generated by multiset grammars are denoted by $m(\mathbf{X})$, $\mathbf{X} \in \{\mathbf{REG}, \mathbf{LIN}, \mathbf{CF}, \mathbf{FIN}, \mathbf{ARB}, \mathbf{MON}, \mathbf{MAT}(\mathbf{MAT}^{\boldsymbol{\lambda}}$- with erasing rule, $\mathbf{MAT}_{\mathbf{ac}}$ - with appearance checking), $\mathbf{LOC}\}$ for regular, linear, context-free, finite, arbitrary, monotone, matrix and local multiset grammars respectively. The languages family of matrix, Parikh sets of vectors and semilinear which denoted by $\mathbf{Ps}(\mathbf{X})$ and $\mathbf{SLin}$ are also considered.

The following theorem present the entrenched relations of families of language generated by multiset grammars.

Theorem 1 [56]

$\mathbf{mFIN} = \mathbf{PsFIN}$. $[a]$
$\mathbf{mREG} = \mathbf{mLIN} = \mathbf{mCF} = \mathbf{PsREG} = \mathbf{PsLIN} = \mathbf{PsCF} = \mathbf{SLin}$. $[b]$
$\mathbf{mMON} = \mathbf{mMAT} = \mathbf{PsMAT}$. $[c]$
$\mathbf{mMAT}_{\mathbf{ac}} = \mathbf{PsMAT}_{\mathbf{ac}}$. $[d]$
$\mathbf{mARB} = \mathbf{mMAT}^{\boldsymbol{\lambda}} = \mathbf{PsMAT}^{\boldsymbol{\lambda}}$. $[e]$
$\mathbf{mMAT}^{\boldsymbol{\lambda}}_{\mathbf{ac}} = \mathbf{PsRE} = \mathbf{PsMAT}^{\boldsymbol{\lambda}}_{\mathbf{ac}}$. $[f]$

From there, we have

$[a] \subset \mathbf{mLOC} \subset [b] \subset [c] \subset [e] \subset [f]$.
$[c] \subset [d] \subset [f]$.

The closure properties of families of language generated by multiset grammars are given in the next theorem.

Theorem 2 [57]

The language family of $\mathbf{mREG}$ is closed under set union, set difference, complement, set intersection, arbitrary homomorphisms, inverse homomorphism, intersection with regular multiset languages, multiset addition and arbitrary substitution.

The language family of $\mathbf{mCF}$ is closed under set union, set intersection, set difference, complement, multiset addition, λ-free homomorphisms, arbitrary homomorphisms, inverse homomorphisms, λ-free substitution, arbitrary substitution and intersection with regular multiset languages.

The language family of $\mathbf{mARB}$ is closed under set union, set interaction, λ-free homomorphisms, arbitrary homomorphisms, inverse homomorphisms, λ-free substitution, arbitrary substitution, multiset union, multiset intersection, multiset difference, multiset addition and intersection with regular multiset languages.

The language family of $\mathbf{mMON}$ is closed under set union, set intersection, λ-free homomorphisms, multiset union, multiset addition, λ-free homomorphisms, inverse homomorphisms, λ-free

substitution, linear erasing, and intersection with regular multiset languages. However, it is not closed under arbitrary homomorphism, arbitrary substitution and multiset difference.

Since then, a plenty of studies have been on multiset grammars in various directions with one inclination to improve its generative power. Therefore, here we continue to review on articles which found federated to multiset grammar since its first discovery until the present one.

In the same year where multiset grammar was discovered, Csuhaj-Varju, Martin and Mitrana [58] immediately came up with a Chomsky hierarchy characterization of multiset grammars in term of multiset automata where their working mode is according to the addition and subtraction of multiset. In that paper [58], they defined three types of multiset automata which are multiset finite automata (MFA), multiset linear bounded automata (MLBA) and multiset Turing machine (MTM).

A multiset finite automata is like an input bag for placing a multiset and a detecting head which used for detecting the existence of a given symbol in that bag. It works as follows: it starts with a multiset in its bag in the initial state where then depending on the former state and a symbol detection in the bag, it will change its current state. A symbol will be automatically eliminated from the bag if it has been located. The automaton will stop when the bag is empty or there are no possible further movement. The input multiset will only be accepted if the bag is empty at the final state [58]. Formally, a multiset finite automata is defined as:

Definition 2 [58] A multiset finite automata is a construction of an automaton having a structure $C = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of non-empty of states, Σ is the alphabet of input, q_0 is the initial (start) state, F is the final state and δ is the mapping of transition

$$\delta : Q \times \Sigma \to 2^Q.$$

A configuration of C is a pair (q, μ) where q is the current state and μ is a multiset. Then, the following relation on all configuration set is defined as $(q, \mu) \Rightarrow (s, \rho)$ iff there exists $a \in \Sigma$ such $s \in \delta(q, a)$ and $\mu - \rho = \mu_a$.

The macroset accepted by C is elucidated by $Rec(C) = \{ \mu \,|(q_0, \mu) \Rightarrow^* (q, \lambda)$ for some $q \in F\}$ with $\Rightarrow^*$ is the reflexive and transitive closure of the operation.

Then, there exists family of all macrosets accepted by

multiset finite automata which denoted by **MFA**

deterministic multiset finite automata which denoted by **DMFA**.

multiset finite automata with detection which denoted by **MFAD**.

deterministic multiset finite automata with detection which denoted by **DMFAD**.

Further, in term of its generative power, some propositions have been proved such :

Proposition 1 [58]

$\mathbf{MFA} = \mathbf{mREG} = \mathbf{mCF} = \mathbf{PsREG} = \mathbf{PsCF} = \mathbf{SLin}$.

DMFA is incomparable with **Lin** and the finite macrosets family respectively to the set inclusion.

$\mathbf{DMFA} \subset \mathbf{MFA}$.

$\mathbf{MFA} = \mathbf{MFAD}$.

DMFAD consists of all finite macrosets.

Lin and **DMFAD** are incomparable respectively to the set inclusion.

Subsequently, a multiset linear bounded automata working procedure is the same as multiset finite automata except in changes state process, it will not only changes its state but it also can add or not a symbol in to the bag contents. Formally, it can be defined as follows.

Definition 3 [58] A multiset linear bounded automaton $(MLBA)$ is a 6-tuples construct such $C = (Q, \Sigma, U, \delta, q_0, F)$ where (Q, Σ, q_0, F) are defined as in MFA, U is the bag alphabets with $\Sigma \subseteq$ U and δ is the the mapping of transition from $Q \times U$ into all subsets set of $Q \times (U \cup \{e\})$. Then, its configuration is a also a pair (q, μ) such q is the present state and μ is the bag content written as $(q, \mu) \Rightarrow (s, \rho)$ if and only if there is $a \in \Sigma$ such

$(s, b) \in \delta(q, a), b \neq e, \mu(a) \geq 1, \rho(a) = \mu(a) - 1,$ $\rho(b) = \mu(b) + 1$, and $\rho(c) = \mu(c)$ for all $c \in \Sigma, c \notin \{a, b\}$ or

$(s, e) \in \delta(q, a), \mu(a) \geq 1, \rho(a) = \mu(a)$ and $\rho(c) = \mu(c)$ for all $c \in \Sigma, c \neq a$.

The macroset accepted by A is expounded by $Rec(A) = \{ \mu \,|(q_0, \mu) \Rightarrow^* (q, \lambda)$ for some $q \in F\}$ with $\Rightarrow^*$ is the reflexive and transitive closure of the opreration.

This type of automaton also have provide some significance propositions and theorem such

Theorem 3 [58] The macrosets class generated by linear bounded grammars is equal to the macrosets class generated by monotone grammars.

Proposition 2 [58]

mMON equals to macrosets class accepted by $MLBAs$.

Deterministic $MLBAs$ are strictly less powerful than $MLBAs$.

Every macroset in multiset random context grammars is accepted by a deterministic $MLBA$.

Another type of multiset automata is multiset Turing machine (MTM) which its working mode is defined precisely like multiset linear bounded automata. However,

unlike multiset linear bounded automata, it has a bag for storing an infinite multiset together with a read-write head which allow a symbol to be picked up from the bag and added the symbol to the bag content at most one symbol. Thus, such machine accepted a multiset if its bag has an infinite number of $W's$ with nothing else and also entered a final state. By denoting β as the multiset $\beta(W) = \infty$ and other symbol a as $\beta(a) = 0$, a multiset turing machine is defined as in the next definition.

Definition 4 [58] A multiset Turing machine is a 7-tuples $T = (Q, \Sigma, U, \delta, q_0, W, F)$ where $(Q, \Sigma, U, q_{0,} F)$ are defined exactly for $MLBA$, W is a special symbol occurs in the bag in infinitely many times and δ is the mapping of transition from $Q \times U$ into all subsets set of $Q \times ((U \cup \{e\})\{W\})$. Then, a configuration of MTM is also written exactly for MLBA but the macroset accepted by T is defined by $Rec(T) = \{ \mu \mid (q_{0,}\mu) \Rightarrow^* (q, \beta)$ for some $q \in F \}$ where its generative powers are demonstrated such

Proposition 3 [58]

The accepting power of $mARB$ is equal to $MTMs$.

Deterministic $MTMs$ are strictly less powerful than $MTMs$.

$MTMs$ with detection are powerful than $MTMs$.

$PsARB$ is equal to the macrosets accepted by $MTMs$ with detection.

Since then, after a while, the direction of multiset study are continuing expanded where in 2007, Cavaliere *et al.* [59] developed a new grammar model known as random context multiset grammars which based on relation of partial order on the objects the grammars contend with together with the multiset random context checkers and transducers concept. In that paper [59], they showed how those grammars can generate set of recursively enumerable of finite multiset and also can be easily enhanced to antiport P system. Further, Wang, Yin and Gu in [60] made use of fuzzy concept to introduce two new extensions of multiset grammars and automata called fuzzy multiset grammars and fuzzy multiset finite automata with discussion of the relationship between fuzzy multiset regular grammars with fuzzy multiset finite automata in 2013. Besides, they defined some closure properties of fuzzy multiset finite languages family under certain regular operations [60]. Shortly thereafter, in 2015, Tiwari, Gautam and Dubey widened the study done by Wang and his friends by associating a deterministic fuzzy multiset finite automaton with a given fuzzy multiset finite automaton and showing that both automata are equally powerful in the sense of fuzzy multiset language acceptance in [61]. They as well studied and presented two minimal realizations of fuzzy multiset language where they proved that both of them are isomorphic [61].

IV. VALENCE GRAMMARS

Valence grammars which is a combination of a grammar with a blind multicounter machine have been acknowledged as one of the favourable controlled grammars in formal language studies area due to their naturalness and simplicity besides having a possession in nice properties concerning closure under operations and problems of decidability. It was independently introduced by Gheorghe Paun in [42] in 1980. His notion was to assign each production with an integer from a given monoid so called valence and then all valences will be added along the applied productions. A derivation is said to be valid if and only if the sum of all valences evaluates to zero or the product of all valences results in one. The formal definition of a valence grammar is defined as follows.

Definition 5 [42] An additive (multiplicative) valence grammar is a 5-tuples $G = (N, T, S, P, v)$ where (N, T, S, P) are defined as for a context-free grammar and v is a mapping from P into $\mathbb{Z}$ ($\mathbb{Q}$). The language generated by the additive (multiplicative) grammar G consists of all string $w \in T^*$ such that there is a derivation $S \xrightarrow{r_1 r_2 \ldots r_n} w$ where

$$\sum_{k=1}^{n} v(r_k) = 0 \left(\prod_{k=1}^{n} v(r_k) = 1 \right).$$

The family of languages generated by additive valence and multiplicative valence grammar are denoted by $\boldsymbol{a}\mathbf{VAL}, \boldsymbol{m}\mathbf{VAL}$ ($\boldsymbol{a}\mathbf{VAL}^{\boldsymbol{\lambda}}, \boldsymbol{m}\mathbf{VAL}^{\boldsymbol{\lambda}}$ – with erasing rule) respectively.

Here, the examples to differentiate between additive valence and multiplicative valence grammar.

Example 1 [53] Let $G = (\{S, A, B\}, \{a, b, c\}, \{p_1, p_2, p_3, p_4, p_5\}, S, v\}$ be an additive valence grammar with $p_1 : S \to AB$, $p_2 : A \to aAb$, $p_3 : B \to cB$, $p_4 : A \to ab$, $p_5 : B \to c$, and $v(p_1) = v(p_4) = v(p_5) = 0$, $v(p_2) = 1$, $v(p_3) = -1$.

Thus, the grammar generates the language $L(G) = \{a^n b^n c^n : n \geq 1\}$.

Example 2 [53] Let $G = (\{S, A, B\}, \{a, b, c\}, \{p_1, p_2, p_3, p_4, p_5\}, S, v\}$ be a multiplicative valence grammar with $p_1 : S \to AB$, $p_2 : A \to aAb$, $p_3 : B \to cB$, $p_4 : A \to ab$, $p_5 : B \to c$, and $v(p_1) = v(p_4) = v(p_5) = 1$, $v(p_2) = 2$, $v(p_3) = \frac{1}{2}$.

Hence, the grammar also generated the same language $L(G) = \{a^n b^n c^n : n \geq 1\}$.

The coming theorems convey the existing of well-established relations of families of languages generated by valence grammars.

Theorem 4 [53,62]

$\mathbf{CF} \subset \boldsymbol{a}\mathbf{VAL} = \boldsymbol{a}\mathbf{VAL}^{\boldsymbol{\lambda}}$ and $\mathbf{CF} \subset \boldsymbol{m}\mathbf{VAL} = \boldsymbol{m}\mathbf{VAL}^{\boldsymbol{\lambda}}$.

$(\boldsymbol{a}\mathbf{VAL}, \mathbf{X}) \subset (\boldsymbol{m}\mathbf{VAL}, \mathbf{X}) = (\boldsymbol{u}\mathbf{V}, \mathbf{X})$ where $\mathbf{X} \in \{\mathbf{REG}, \mathbf{CF}, \mathbf{CF}^{\lambda}, \mathbf{CS}\}$.

The closure properties and decidability problems of valence grammars are given in the following theorems.

Theorem 5 [53, 62]

The family of language of $\boldsymbol{a}\mathbf{VAL}$ is closed under union, intersection by regular sets, arbitrary morphisms, λ-free morphisms, inverse morphisms, λ-free gsm-mappings, gsm-mappings, quotient by regular sets and quotient by letters but it is not closed under intersection, complementation, concatenation and kleene-closure.

The family of language of $\boldsymbol{m}\mathbf{VAL}$ is closed under union, concatenation, intersection by regular sets, gsm-mappings, λ-free gsm-mappings, arbitrary morphisms, λ-free morphisms, inverse morphisms, quotient by regular sets and quotient by letters. However, it is not closed under intersection, complementation and Kleene-closure.

Theorem 6 [53] The membership, emptiness and finiteness problems can be decided in both families of languages of $\boldsymbol{a}\mathbf{VAL}$ and $\boldsymbol{m}\mathbf{VAL}$.

In 1997, Fernau and Stiebe [43] continued investigating the closure properties of languages of valence and simultaneously they showed that valence grammars over arbitrary monoids can stimulate equivalent matrix grammars. They also demonstrated that by constructing valence grammars over $(\mathbb{Q}_+, \times, 1)$, there exist normal forms for unordered vector grammars. In the meantime, they have examined the use of valences in parallel systems [43].

Shortly thereafter, the same authors in [45] extended the concept of valence grammar by the notion like a valid derivation value is acceptable to be a part of a given target set. In that paper, they also have studied the closure properties of those grammars together with their generative power with target sets over the groups $\mathbb{Z}^k$, monoids $\mathbb{N}^k$, and finite monoid. Furthermore, they proved that grammars with permutations of regular languages as a controller can characterize the unordered vector languages and at the same time, demonstrated that valence grammars with finite monoids as target sets can be constructed into an equivalent matrix language [45]. Interestingly, in the same year, Stiebe alone in [47] came out with a notion such by assessing the productions of grammar with integer vectors, we can generate an equivalent matrix grammar as well as Parikh languages. Those grammar is called positive valence grammar.

Then, one year later, Fernau and Stiebe together again studied the power of use of valence in sequential grammar and automata in [44] where then they showed a procedure on how to construct the Chomsky and Greibach normal form for those grammars. Besides, in that paper, they also proved that context free valence grammars over commutative or finite monoids have an equivalent power as valence grammars over commutative groups or finite group respectively. Later, after a few years which is in 2008, Render and Kambites [41] continued the study done by [45], where they investigate the languages class recognized by polycyclic and bicyclic valence automata (or identically regular valence grammar) with rational target as well as together with the closure properties and rational subset membership decidability problem of those grammar where as their main results they showed that such automata have accepted exactly the languages of context-free for the case polycyclic monoids of rank two or more and the languages class which including the languages of partially blind one counter for the case bicyclic monoids (polycyclic monoid of rank 1) [41].

The study concerning with the accepting power of valence automata does not standstill there where in 2013, Buckheister and his Zetzsche in [46] persisted to study which monoids valence automata can recognize only the languages of context-free and also the languages in the company of semilinear Parikh image, respectively. In those [46], they demonstrated a characterization of monoids graph products for valence automata to accept only the languages of context-free and also for the bicyclic monoids and integers graph product to yield only the languages which with semilinear Parikh image. They also proved that all languages acknowledged by valence automata over torsion groups will possess a semilinear Parikh image [46].

V. TREE CONTROLLED GRAMMARS

The idea of imposing restrictions upon the derivation trees of context-free grammars was originated by Culik and Maurer in [30] where they introduced a new regulated grammar called tree controlled grammars (for short TC grammars) in 1977. In this section, we present an indigenous definition of a tree controlled grammar defined in [30] together with an example and its well-proven generative power in [53] [63] [31] as well as along with a review on a continued studies done which found concomitant with it.

Definition 6 [30] A tree controlled grammar is a quintuple $G = (N, T, P, S, R)$ where (N, T, P, S) is defined exactly as a context-free grammar and $R \subseteq (N \cup T)^*$ is a regular set. It can also be considered as a pair $G' = (G, R)$. Then, the language $L(G)$ consists of all words w generated by the underlying grammar G in such a way that there exists a derivation tree t of w with respect to G, where the concatenating of words at any level of tree from left to right excluding the last one are in R. The families of all tree

controlled grammars without or with erasing rules are denoted by a symbol **TC** and $\mathbf{TC}^{\lambda}$.

To demonstrate the working principle of a tree controlled grammar, we consider the following example.

Example 3 [30] Let $G' = (G, R)$ where $G = (\{S\}, \{a\}, \{S \to SS|a\}, S\})$ and $R = \{S\}^*$. For the control language R consists of sequences of symbol of nonterminal S known as word w with $w \in L(G')$ where all the nodes in every level of the derivation tree except the last one are labelled by S. This means, at every level of tree which not including the last one, we applied the production rule as $S \to SS$ in all cases and then $S \to a$ at the last level since w is the word contains terminal symbols only. Hence, G' generated the language $L(G') = \{a^{2^n} | a \geq 0\}$.

There are numerous of momentous theorems proved in TC grammars such in term of its

characteristics, we have the theorems like

Theorem 7 [30] There is an algorithm for TC grammars which working in time $O(n^2)$ for any word w with $|w| = n$ if $w \in L(G, R)$ and G is unambiguous.

Theorem 8 [30] In every TC grammars, the language $L(G, R)$ is recursive when there is no empty word at all on the right side of its production.

computational power, we have the theorem such

Theorem 9 [30] The families of languages **REG**, **LIN**, **CF**, **RE**, **EOL** and **ETOL** can be generated by TC grammars in an innate manner.

Since the TC grammar was properly defined, a great extent of researches have been done on it in variety directions with a desire to further empower its generative capacity where soon after which is in 1979, Paun in [63] thoroughly examined the generative power of a TC grammar by not only considering the context-free grammar controlled by regular languages but by considering all possible variants of the TC grammar with varying the grammars and its control language its. Thus, he extends the definition of TC grammars as in Definition 7.

Definition 7 [63] A tree controlled grammar is considered a pair (G, M) instead of (G, R) as in [30] where G is a grammar of type-i such $G = (N, T, P, S)$ and M is a language type-j such $M \subset (N \cup T)^*$ with $i = \mathbf{CF}, \mathbf{CF}^{\lambda}, \mathbf{REG}$ and $j = \mathbf{RE}, \mathbf{CS}, \mathbf{CF}, \mathbf{REG}, \mathbf{FIN}$. Then, the language generated by (G, M) is written as $L(G, M)$ with condition such there exists a derivation tree t of words w with respect to G, where the concatenating of the words from left to right at any level of tree excluding the last one are in M. The families of languages of TC grammars without or with erasing rules are denoted by a symbol $\mathbf{TC}(\mathbf{i}, \mathbf{j})$ and $\mathbf{TC}^{\lambda}(\mathbf{i}, \mathbf{j})$.

From those definition, the following theorems are obtained.

Theorem 10 [63]

$\mathbf{TC}(\mathbf{REG}, \mathbf{j}) = \mathbf{REG}$ for $\mathbf{j} = \mathbf{RE}, \mathbf{CS}, \mathbf{CF}, \mathbf{REG}, \mathbf{FIN}$.
$\mathbf{TC}(\mathbf{CF}^{\lambda}, \mathbf{j}) = \mathbf{RE}$, for $\mathbf{j} = \mathbf{RE}, \mathbf{CS}, \mathbf{CF}, \mathbf{REG}$.
$\mathbf{TC}(\mathbf{CF}, \mathbf{RE}) = \mathbf{RE}$.
$\mathbf{TC}(\mathbf{CF}, \mathbf{j}) = \mathbf{CS}$, for $\mathbf{j} = \mathbf{CS}, \mathbf{CF}, \mathbf{REG}$.
$\mathbf{TC}(\mathbf{i}, \mathbf{FIN}) = \mathbf{MAT}, \mathbf{i} = \mathbf{CF}, \mathbf{CF}^{\lambda}$.

Then, after around twenty years elapsed, there has arisen an issue whether there is a possibility for a TC grammar to possess the same power as it is if its derivation tree level is controlled by subregular languages. This problem has been investigated by Dassow and Truthe in [31] in 2008 where they considered several different types of subregular languages such as finite, combinational, monoids, regular suffix-closed, nilpotent, non-counting, regular commutative and circular languages. Those grammars are called "subregularly tree controlled grammars" and were defined as in the next theorem.

Definition 8 [31] A subregularly tree controlled grammars is a quintuple $G = (N, T, P, S, R)$ where (N, T, P, S) is defined as a context-free grammar and R belongs to some special subfamily of regular languages family such regular circular, combinational, regular suffix-closed, definite, regular commutative, regular power-separating, regular non-counting, nilpotent and ordered which denoted by **CIRC**, **COMB**, **SUF**, **DEF**, **COMM**, **PS**, **NC**, **NIL**, and **ORD**. The language $L(G)$ contains all words w generated by the underlying grammar G in such a way there exists a derivation tree t of w with respect to G, where the words of all levels (except the last one) are in R. The families of all subregularly tree controlled grammars are denoted by $\mathbf{TC}(\mathbf{X}), \mathbf{X} \in \{\mathbf{CIRC}, \mathbf{COMB}, \mathbf{SUF}, \mathbf{DEF}, \mathbf{ORD}, \mathbf{COMM}, \mathbf{PS}, \mathbf{NC}, \mathbf{NIL}\}$ ($\mathbf{TC}^{\lambda}(\mathbf{X})$ – with erasing rule).

The subregularly TC grammars have achieved a good result in generative power as showed in subsequent theorems.

Theorem 11 [31]

$\mathbf{RE} = \mathbf{TC}^{\lambda}(\mathbf{REG}) = \mathbf{TC}^{\lambda}(\mathbf{SUF}) = \mathbf{TC}^{\lambda}(\mathbf{ORD}) = \mathbf{TC}^{\lambda}(\mathbf{NC}) = \mathbf{TC}^{\lambda}(\mathbf{PS}) = \mathbf{TC}^{\lambda}(\mathbf{COMM}) = \mathbf{TC}^{\lambda}(\mathbf{CIRC})$.
$\mathbf{TC}(\mathbf{COMM}) = \mathbf{MAT}$.
$\mathbf{EOL} = \mathbf{TC}(\mathbf{MON}) \subseteq \mathbf{TC}(\mathbf{COMB}) \subseteq \mathbf{TC}(\mathbf{DEF})$.
$\mathbf{EOL} \subset \mathbf{TC}(\mathbf{DEF})$.
$\mathbf{TC}(\mathbf{FIN}) \subset \mathbf{TC}(\mathbf{NIL})$ and $\mathbf{TC}(\mathbf{MON}) \subset \mathbf{TC}(\mathbf{NIL})$.
$\mathbf{MAT}^{\mathbf{FIN}} = \mathbf{TC}^{\lambda}(\mathbf{FIN}) \subset \mathbf{TC}^{\lambda}(\mathbf{NIL}) \subseteq$ and
$\mathbf{TC}^{\lambda}(\mathbf{MON}) \subset \mathbf{TC}^{\lambda}(\mathbf{NIL})$.

$\mathbf{CF} \subset \mathbf{EOL} = \mathbf{TC}^{\lambda}(\mathbf{MON}) \subseteq \mathbf{TC}^{\lambda}(\mathbf{COMB}) \subseteq \mathbf{TC}^{\lambda}(\mathbf{DEF}) \subseteq \mathbf{RE}$ and $\mathbf{TC}^{\lambda}(\mathbf{MON}) \subset \mathbf{TC}^{\lambda}(\mathbf{DEF})$.

The investigation of subregularly TC grammars does not stop there where in the same year in [32] by Dassow and Truthe again continued to study on the hierarchy of subregularly TC languages. In that paper, 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 automata with mostly prescribed number of states. This is the starting where the complexity of tree controlled grammars are being investigated.

Then, in the same year as [31] also, [35] validated that every linearly bounded queue automaton has a TC grammar. He also showed that 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 and if erasing productions are allowed in the grammar, it can generate the recursively enumerable languages [35]. Subsequently, Turaev, Dassow and Selamat in [37] recommenced investigating the TC grammars in the company of bounded nonterminal complexity in 2011. 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 [37]. The same authors, Turaev et al. in another paper in [36] 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 [36].

Later on, in 2012, by using the same technique as in [36] but with different version of the Geffert normal form, Vaszil presented that the complexity of nonterminal of TC grammars can be reduced from seven to six in [34]. Interestingly, in the same year, Koutny and Meduna in [33] came out with a new different idea of generating TC grammars where instead of placing the restrictions on tree levels, they placed them on 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 [33].

VI. CONCLUSION AND FUTURE WORK

In a nut shell, we have designated a review on the topics related to arithmetically controlled grammars and tree controlled grammars, such as multiset grammars, valence grammars and tree controlled grammars. From the original definition of those three controlled grammars, we can see that all of them have a simple procedure in term of generating the languages yet they are powerful grammars since they have achieved many remarkable result in formal language theory.

However, there are still some captivating topics in this direction to look for future study. First, if we notice, in the multiset grammar review, there is no research done in using multiset on terminal symbols which can be based on an operation namely "counter" where in every production in the grammar, a multiset value will be given to it depending on the number of terminal alphabet existed in the right hand side of that production as a control mechanism. For example, if a production has one terminal symbol "a" and two terminal symbol "b", its multiset will be counting as "1" for "a" and "2" for "b" in vector form. Then, a derivation in the grammar is called successful when it satisfied certain function.

Other than that, it will be more interesting if we combine the valence and tree concepts to control the derivation of grammar like in tree controlled grammars. In this way, we can replace the 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 one represented in matrices form in which the sum of those values given in the value of main production. Then, a derivation in grammar is called a successful one if and only if there are such permutations in each row yield a value of zero. This idea also can be combined with multiset rather than valence by replacing the regular set with certain summation of natural number. Besides, rather than checking the production rules with regular set, we can implement regular sets of production rules of the grammars.

REFERENCES

[1] A.M. Turing, "On Computable Numbers with and An Application to the Entscheidungsproblem," in *Proceeding of the London Mathematical Society*, 1937.

[2] N. Chomsky, "Three Models for the Description of Language," *IRE Transaction, Information Theory*, vol. 2, no. 3, pp. 113-124, 1956.

[3] S. Ginsburg and H. Rice, "Two Families of Languages Related to Algol.," *Journal of the ACM*, vol. 9, no. 3, pp. 350-371, 1962.

[4] A. Salomaa, Formal Language, New York: New York Academic Press, 1973.

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

[6] G. Bel-Enguix, M. Jiménez-López and C. Martín-Vide, New Developments in Formal Languages and Applications, vol. 113, Springer Berlin Heidelberg, 2008.

[7] A.M. Natarajan, A. Tamilarasi and P. Balasubramani, Theory of Automata and Formal Languages, New Delhi: New Age International (p) Ltd, 2008.

[8] P. Linz, An Introduction to Formal Languages and Automata, Massachusetts: Jones and Bartlett, 2001.

[9] C. Martín-Vide, V. Mitrana and G. Paun, Formal Languages and Applications, vol. 148, Springer Berlin Heidelberg, 2004.

[10] J. Dassow, G. Paun and A. Salomaa, "Grammars with Controlled Derivations," *Handbook of Formal Languages,* vol. 2, pp. 101-154, 1997.

[11] A. Abraham, "Some Questions of Phrase Structure Grammars," *Computational Linguistics,* vol. 4, pp. 61-70, 1965.

[12] A. Cremers and O. Mayer, "On Matrix Languages," *Information and Control,* vol. 23, pp. 86-96, 1973.

[13] S. Abraham, "Some Questions of Language Theory," in *Proceeding COLING '65 Proceedings of the 1965 conference on Computational linguistics*, Bonn, Germany, 1965.

[14] O.H. Ibarra, "Simple Matrix Languages," *Information and Control,* vol. 17, pp. 359-394, 1970.

[15] G. Paun, "On The Generative Capacity of Simple Matrix Grammars of Finite Index," *Information Processing Letters,* vol. 7, no. 2, pp. 100-102, 1978.

[16] G. Paun, "On the Family of Finite Index Matrix Languages," *Journal of Computer and System Sciences,* vol. 18, pp. 267-280, 1979.

[17] A. Salomaa, "Matrix Grammars with a Leftmost Restriction," *Information and Control,* vol. 20, pp. 143-149, 1972.

[18] S. Ginsburg and E.H. Spanier, "Control Sets on Grammars," *Mathematical Systems Theory,* vol. 2, no. 2, pp. 159-177, 1968.

[19] A. Cremers and O. Mayer, "On Vector Languages," *Journal of Computer and System Sciences,* vol. 8, no. 2, pp. 158-166, 1974.

[20] J. Dassow and S. Turaev, "Petri net Controlled Grammars:the Power of Labelling and Final Markings," vol. 12, no. 2, pp. 191-207, 2009.

[21] J. Dassow and S. Turaev, "Petri net Controlled Grammars:the Case of Special Petri Nets," *Journal of Universal Computer Science,* vol. 15, no. 12, pp. 2808-2835, 2009.

[22] J. Dassow and S. Turaev, "k-Petri Net Controlled Grammars," *Language and Automata Theory and Application,* vol. 5196, pp. 209-220, 2008.

[23] N. Jan, S. Turaev, W. Fong and N. Sarmin, "A new variant of Petri net controlled grammars," in *AIP Conference Proceedings 1682, pp. 040015*, 2015.

[24] S. Turaev, Petri net Controlled Grammars. PHD Thesis, University Rovira i Virgili, 2010.

[25] G. Mavlankulov, M. Othman, M. Selamat and S. Turaev, "Concurrent Context-free Grammars," in In: H. Tutut, D.M. Mat, A. Jemal (Eds.), DaEng - 2013, Lecture Notes in Electrical Engineering, 285,, vol. 285, Springer-Verlag, Berlin, 2014, pp. 521-528.

[26] R. Stiebe and S. Turaev, "Capacity Bounded Grammars," Journal of Automata, Languages and Combinatorics, vol. 15, no. 1/2, pp. 175-194, 2010.

[27] R. Stiebe, "On Grammars Controlled by Parikh Vectors," Languages Alive, vol. 7300, pp. 246-264, 2012.

[28] I. Fris, "Grammars with Partial Ordering of the Rules," Information and Control, vol. 12, no. 5, pp. 415-425, 1968.

[29] A. Cremers, H. Maurer and O. Mayer, "A Note on Leftmost Restricted Random Context Grammars," Information Processing Letters, vol. 2, no. 2, pp. 31-33, 1973.

[30] K. Culik-II and H.Maurer, "Tree Controlled Grammars," Computing, vol. 19, no. 2, pp. 129-139, 1977.

[31] J. Dassow and B. Truthe, "Subregularly Tree Controlled Grammars and Languages," in Proceeding in E. Csuhaj-Varju, Z.Esik (Eds): Automata and Formal Languages, Balatonfured, Hungary, 2008.

[32] J. Dassow and B. Truthe, "On Two Hierarchies of Subregularly Tree Controlled Languages," in Proceeding in C. Campeanu, G. Pighizzini (eds): Descriptional Complexity of Formal Systems, Charlottetown PE, Canada, 2008.

[33] J. Koutny and A. Meduna, "Tree Controlled Grammars with Restriction Placed Upon Cuts and Paths," Kybernetika, vol. 48, no. 1, pp. 165-175, 2012.

[34] G. Vaszil, "On the Nonterminal Complexity of Tree Controlled Grammars," Languages Live, vol. 7300, pp. 265-272, 2012.

[35] R. Stiebe, "On the Complexity of the Control Language in Tree Controlled Grammars," in Proceeding in J. Dassow, B. Truthe (Eds): Colloquium on the Occasion of the 50th Birthday of Victor Mitrana, Otto von Guericke Universitat Magdeburg, Germany, 2008.

[36] S. Turaev, J. Dassow and M.H. Selamat, "Language Classes Generated by Tree Controlled Grammars with Bounded Nonterminal Complexcity," Descriptional Complexity of Formal Systems, vol. 6808, pp. 289-300, 2011.

[37] S. Turaev, M.H. Selamat and J. Dassow, "Nonterminal Complexity of Tree Controlled Grammars," Theoretical Computer Science, vol. 412, no. 41, pp. 5789-5795, 2011.

[38] J. Kelemen, "Conditional grammars: Motivations, definition and some properties," in Proc. Conf. on Automata, Languages and Mathematical Systems, Salgótarjan, Hungary, 1984.

[39] D. Lobo, F. Vico and J. Dassow, "Graph Grammars with String Regulated Rewriting," Theoretical Computer Science, vol. 412, no. 43, pp. 6101-6111, 2011.

[40] D.J. Rosenkrantz, "Programmed Grammars and Classes of Formal Languages," Journal of the ACM, vol. 16, no. 1, pp. 107-131, 1969.

[41] E. Render and M. Kambites, "Polycyclic and Bicyclic Valence Automata," Language and Automata Theory and Applications, vol. 5196, pp. 464-475, 2008.

[42] G. Paun, "A New Generative Device: Valence Grammars," Revue Roumaine de Mathematiques Pures et Appliquees, vol. 6, pp. 911-924, 1980.

[43] H. Fernau and R. Stiebe, "Regulation by Valences," Mathematical Foundations of Compuer Science, pp. 239-248, 1997.

[44] H. Fernau and R. Stiebe, "Sequential Grammar and Automata with Valences," Theoretical Computer Science, vol. 276, pp. 377-405, 2002.

[45] H. Fernau and R. Stiebe, "Valence Grammars with Target Sets," in Proceding in S. Yu, M. Ito, Gh. Paun (Eds) : Words, Semigroups and Transductions, World Scientific, Singapore, 2001.

[46] P. Buckheister and G. Zetzsche, "Semilinearity and Context-Freeness of Languages Accepted by Valence Automata," Mathematical Foundations of Computer Science 2013, vol. 8087, pp. 231-242, 2013.

[47] R. Stiebe, "Positive Valence Grammars," in In Proceeding: J.Dassow und Bernd Reichel (Hrsg), Theorietag Automaten und Formale Sprachen,Coding Theory and Formal Languages, Wendgraben, 2001.

[48] A. Aho, "Indexed Grammar : An extension of Context-Free Grammars," Journal of the ACM, vol. 15, no. 4, pp. 647-671 , 1968.

[49] S. Greibach and J. Hopcroft, "Scattered context grammars," Journal of Computer and System Sciences, vol. 3, no. 3, pp. 233-247, 1969.

[50] M. Levitina, "On Some Grammars with Global Productions.," NTI Ser., vol. 2, no. 3, pp. 32-36, 1972.

[51] R. Siromoney and K. Krithivasan, "Parallel Context-Free Languages," Information and Control, vol. 24, no. 2, pp. 155-162, 1974.

[52] J.M. Castano, "Global Index Grammars and Descriptive Power,"

Journal of Logic, Language and Information, vol. 13, no. 4, p. 403–419, 2004.

[53] J. Dassow and G. Paun, Regulated Rewriting in Formal Language Theory, vol. 18, Springer-Verlag Berlin Heidelberg, 1989.

[54] W.D. Blizard, "Multiset Theory," Notre Dame Journal of Formal Logic, vol. 30, no. 1, pp. 36-66, 1989.

[55] D. Singh, A. Ibrahim, T. Yohanna and J. Singh, "A systematization of Fundamentals of Multisets," Lecturas Matematicas, vol. 29, pp. 33-48, 2008.

[56] M. Kudlek, C. Martin-Vide and G. Paun, "Toward a Formal Macroset Theory," Multiset Processing, vol. 2235, pp. 123-133, 2001.

[57] M. Kudlek and V. Mitrana, "Closure Properties of Multiset Language Families," Membrane Computing, Fundamenta Informatica, vol. 49, no. 1, pp. 191-203, 2002.

[58] E. Csuhaj-Varju, C. Martin-Vide and V. Mitrana, "Multiset Automata," Multiset Processing, vol. 2235, pp. 69-83, 2001.

[59] M. Cavaliere, R. Freund, M. Oswald and D. Sburlan, "Multiset Random Context Grammars, Checkers, and Transducers," Theoretical Computer Science, vol. 372, no. 2-3, pp. 136-151, 2007.

[60] J. Wang, M. Yin and W. Gu, "Fuzzy Multiset Finite Automata and Their Languages," Soft Computing, vol. 17, no. 3, pp. 381-390, 2013.

[61] S. Tiwari, V. Gautam and M. Dubey, "On Fuzzy Multiset Automata," Journal of Applied Mathematics and Computing, vol. 51, no. 1, pp. 643-657, 2015.

[62] J. Dassow, "Grammars With Regulated Rewriting," Formal Languages and Applications, vol. 148, pp. 249-273, 2004.

[63] G. Paun, "On the Generative Capacity of Tree Controlled Grammars," Computing, vol. 21, no. 3, pp. 213-220, 1979.