RESTRICTIONS ON DERIVATIONS IN *N*-GENERATING GRAMMAR SYSTEMS

Martin Čermák
Doctoral Degree Programme (IV), FIT BUT
E-mail: icermak@fit.vutbr.cz

Supervised by: Alexaner Meduna
E-mail: meduna@fit.vutbr.cz

Abstract: From a pragmatic point of view, an investigation of various type of restrictions, placed on derivation in grammars, plays a significant role in the formal language theory. The main reason is a possibility to select nonterminals intended to rewrite with a lower level of nondeterminism. This paper introduces generalized version of *n*-generating grammar system and corresponding *n*-accepting move-restricted automata system. On the *n*-generating grammar system, it shows how two variants of derivation-restrictions effect its generative power.

Keywords: automata system, grammar system, *n*–language, control computation, leftmost derivation

1 INTRODUCTION

The formal language theory has investigated various type of left restrictions in grammars working in a context-free way. The motivation for this investigation comes from pragmatical usage of grammars, where selection of rewriting nonterminals with a lower level of nondeterminism are needed. In ordinary context-free grammars, these restrictions usually have no effect on the generative power. However, in terms of regulated context-free grammars, the left derivation restrictions have an effect on their generative power (see [1, 2]).

The authors of [6] studied general and canonical multi-generative grammar systems, where the main difference between them is leftmost restriction during generating placed on derivations in all grammars used in the system. In [6] it has been proven that each component in the systems can generate **RE** languages when all the components works in the leftmost way even if the system contains only two context-free grammars. As a corollary we can say, that two context-free grammars making leftmost derivation is enough to give **RE** languages from each components. It does not matter if the restriction is placed on the other components or not. In [5] it has been proven, that general multi-generative systems have only the same generative power as a matrix grammars with epsilon rules. In this paper, effect of differently restricted context-free components in *n*-generative grammar systems are studied.

2 PRELIMINARIES

In this paper, we assume the reader is familiar with the formal language theory (see [7]).

For a set, Q, $|Q|$ denotes the cardinality of Q. For an alphabet, V, V^* represents the free monoid generated by V (under the operation concatenation). The identity of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V. For every string $w \in V^*$, $|w|$ denotes the length of w and w^R denotes the mirror image of w.

A *finite automaton*, FA, is a five-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is an alphabet, $q_0 \in Q$ is the initial state, δ is a finite set of transition rules of the form $qa \to p$, where $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $F \subseteq Q$ is a set of final states. A configuration of M is any string from $Q\Sigma^*$.

For any configuration qay, where $a \in \Sigma$, $y \in \Sigma^*$, $q \in Q$ and any $qa \to p \in \delta$, M makes a move from configuration qay to configuration py according to $qa \to p$, written as $qay \Rightarrow py\,[qa \to p]$, or simply $qay \Rightarrow py$. $\Rightarrow^*$ and $\Rightarrow^+$ represent transitive-reflexive and transitive closure of $\Rightarrow$, respectively. If $w \in \Sigma^*$ and $q_0w \Rightarrow^* f$, where $f \in F$, then w is accepted by M and $q_0w \Rightarrow^* f$ is an acceptance of w in M. The language of M is defined as $\mathcal{L}(M) = \{w \in \Sigma^*\colon q_0w \Rightarrow^* f$ is an acceptance of $w\}$.

A *pushdown automaton*, PDA, is a septuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$, where Q is a finite set of states, Σ is an alphabet, $q_0 \in Q$ is the initial state, Γ is a pushdown alphabet, δ is a finite set of transition rules of the form $Zqa \to \gamma p$, where $p, q \in Q$, $Z \in \Gamma$, $a \in \Sigma \cup \{\varepsilon\}$, $\gamma \in \Gamma^*$ and $Z_0 \in \Gamma$ is the initial pushdown symbol. A configuration of M is any string from $\Gamma^*Q\Sigma^*$. For any configuration $xAqay$, where $x \in \Gamma^*$, $y \in \Sigma^*$, $q \in Q$ and any $Aqa \to \gamma p \in \delta$, M makes a move from configuration $xAqay$ to configuration $x\gamma py$ according to $Aqa \to \gamma p$, written as $xAqay \Rightarrow x\gamma py\,[Aqa \to \gamma p]$, or simply $xAqay \Rightarrow x\gamma py$. $\Rightarrow^*$ and $\Rightarrow^+$ represent transitive-reflexive and transitive closure of $\Rightarrow$, respectively. If $w \in \Sigma^*$ and $Z_0q_0w \Rightarrow^* f$, where $f \in Q$, then w is accepted by M and $Z_0q_0w \Rightarrow^* f$ is an acceptance of w in M. The language of M is defined as $\mathcal{L}(M) = \{w \in \Sigma^*\colon Z_0q_0w \Rightarrow^* f$ is an acceptance of $w\}$.

A *1-turn PDA* is a PDA in which the length of the pushdown tape alternatively increases and decreases at most one times during any sweep of the automation.

A *state-less PDA*, SPDA, is a PDA with exactely one state.

A *Turing machine*, TM, is a finite automaton with infinite tape containing an input string followed by infinite sequence of blank-symbols. The machine has a read-write head which is scanning a single cell on the tape. This read-write head, with respect of rules of the machine, can move left and right along the tape to scan and rewrite successive cells. The string written at the beginning on the input tape is accepted, if the machine moves to finial state.

A *linear bounded automaton*, LBA, is a TM with limited size of tape by the length of its input string.

A *context-free grammar*, CFG, is quadruple $G = (N, T, S, P)$, where N and T are disjoint alphabets of nonterminal and terminal symbols, respectively, $S \in N$ is the start symbol of G, and P is finite set of grammar rules of the form $A \to \alpha$, where $A \in N$ and $\alpha \in (N \cup T)^*$. Let $u, v \in (N \cup T)^*$, for all $r{:}A \to \alpha \in P$, we write $uAv \Rightarrow u\alpha v[r]$, or simply $uAv \Rightarrow u\alpha v$. Let $\Rightarrow^*$ and $\Rightarrow^+$ denote transitive-reflexive and transitive closure of $\Rightarrow$. The language of G is defined as $L(G) = \{\omega\colon S \Rightarrow^* \omega, \omega \in T^*\}$. CFG is *linear*, LNG, and *right-linear*, RLNG, if all its rules are of the form $A \to \alpha$ with $\alpha \in (T^*NT^*) \cup T^*$ and $\alpha \in TN \cup T \cup \{\varepsilon\}$, respectively.

A *valence context-free grammar*, VAL, is quadruple $G = (N, T, S, P)$, where N, T, and S are defined as in CFG, P is finite set of grammar rules of the form $A \to \alpha(n_1, \ldots, n_k)$, where $A \in N$ and $\alpha \in (N \cup T)^*$, $(n_1, \ldots, n_k) \in \mathbb{Z}^k$. At the beginning of the generation the grammar has valence-vector $v \in \mathbb{N}_0^k$ set to $(0, \ldots, 0)$. Let $u, v \in (N \cup T)^*$, for all $r{:}A \to \alpha(n_1, \ldots, n_k) \in P$. Then, $uAv \Rightarrow u\alpha v[r]$, or simply $uAv \Rightarrow u\alpha v$, and $(n_1, \ldots, n_k)$ is added to v. Let $\Rightarrow^*$ and $\Rightarrow^+$ denote transitive-reflexive and transitive closure of $\Rightarrow$. The language of G is defined as $L(G) = \{\omega\colon S \Rightarrow^* \omega, \omega \in T^*$ and valence vector $v = (0, \ldots, 0)\}$.

3 HYBRID CANONICAL RULE–SYNCHRONIZED *N*-GENERATIVE GRAMMAR SYSTEM

A hybrid canonical rule-synchronized multigenerative grammar system consists of n generative components (grammars) and a control set of n-tuples of rules. Each grammar generates its own string in the leftmost way, while the control set determines which grammar rules can be used at the same computation step in all components. In this paper we consider combination of right-linear, linear, and context-free grammars. Formally, the *hybrid canonical rule–synchronized multigenerative grammar system*, HCGR$^{(t_1,\ldots,t_n)}$, is an n+1-tuple $\Gamma = (G_1, \ldots, G_n, Q)$, where $G_i = (N_i, T_i, P_i, S_i)$ is a right-linear, linear, or context-free grammar for every $i = 1, \ldots, n$, and Q is a finite set of n-tuples of the form $(r_1, \ldots, r_n)$, where $r_i \in P_i$ for every $i = 1, \ldots, n$. Furthermore, for all $i = 1, \ldots, n$,

$t_i \in \{$RLNG, LNG, CFG$\}$ denotes type of ith component.

A sentential n-form of HCGR$^{(t_1,\dots,t_n)}$ is an n-tuple $\chi = (x_1,\dots,x_n)$, where $x_i \in (N_i \cup T_i)^*$ for all $i = 1,\dots,n$. Consider two sentential n-forms, $\chi = (u_1A_1v_1,\dots, u_nA_nv_n)$ and $\chi' = (u_1x_1v_1,\dots,u_nx_nv_n)$ with $A_i \in N_i$, $u_i \in T^*$, $v_i,x_i \in (N \cup T)^*$, $r_i{:}A_i \to x_i \in P_i$ for all $i = 1,\dots,n$, and $(r_1,\dots,r_n) \in Q$. Then, $\chi \Rightarrow \chi'$, and $\Rightarrow^*$ and $\Rightarrow^+$ are its transitive-reflexive and transitive closure, respectively. *The n-language of* Γ is defined as n-$L(\Gamma) = \{(w_1,\dots,w_n){:}\ (S_1,\dots,S_n) \Rightarrow^* (w_1,\dots,w_n), w_i \in T_i^*$ for all $i = 1,\dots,n\}$.

4 HYBRID N-ACCEPTING RESTRICTED AUTOMATA SYSTEMS

Formaly, a hybrid *n-accepting move-restricted automata system*, denoted HMAS$^{(t_1,\dots,t_n)}$, is defined as an $n+1$-tuple $\vartheta = (M_1\dots,M_n,\Psi)$ with M_i as a finite or (1-turn) pushdown automaton for all $i = 1,\dots,n$, and with Ψ as a finite set of n-tuples of the form $(r_1,\dots,r_n)$, where for every $j = 1,\dots,n$, $r_j \in \delta_j$ in M_j. Furthermore, for all $i = 1,\dots,n$, $t_i \in \{$FA, 1-turn PDA, PDA$\}$ indicates the type of ith automaton.

An n-configuration is defined as an n-tuple $\chi = (x_1,\dots,x_n)$, where for all $i = 1,\dots,n$, x_i is a configuration of M_i. Let $\chi = (x_1,\dots,x_n)$ and $\chi' = (x'_1,\dots,x'_n)$ be two n-configurations, where for all $i = 1,\dots,n$, $x_i \Rightarrow x'_i\,[r_i]$ in M_i, and $(r_1,\dots,r_n) \in \Psi$, then ϑ moves from n-configuration χ to n-configuration χ', denoted $\chi \Rightarrow \chi'$, and in the standard way, $\Rightarrow^*$ and $\Rightarrow^+$ denote the transitive-reflexive and the transitive closure of $\Rightarrow$, respectively.

Let $\chi_0 = (x_1\omega_1,\dots,x_n\omega_n)$ be the start and $\chi_f = (q_1,\dots, q_n)$ be a final n-configuration of HMAS$^{(t_1\dots,t_n)}$, where for all $i = 1,\dots,n$, ω_i is the input string of M_i and q_i is state of M_i. The n-language of HMAS$^{(t_1,\dots,t_n)}$ is defined as n-$L(\vartheta) = \{(\omega_1,\dots,\omega_n){:}\ \chi_0 \Rightarrow {}^*\chi_f$ and for every $i = 1,\dots,n$, M_i accepts$\}$.

In a special case where all components are of type X, we write ${}_nX$ instead of $(X,\dots,X)$. For example, a hybrid n-accepting move-restricted automata system, where all components are PDAs, is denoted by HMAS$^{{}_nPDA}$.

5 PROPERTIES OF N-LANGUAGES GIVEN BY RESTRICTED N-SYSTEMS

First of all we placed the leftmost restriction on derivations in the first component. Derivations of the others are not restricted.

Theorem 1. *Let $\mathscr{L}(HCGR^{(CFG_{lm},CFG)})_1$ is the family of languages generated by the first component working in the leftmost way in $HCGR^{(CFG_{lm},CFG)}$ and $\mathscr{L}(VAL_{lm},\mathbf{CF})$ is the family of leftmost valence context-free grammar. Then, $\mathscr{L}(HCGR^{(CFG_{lm},CFG)})_1 = \mathscr{L}(VAL_{lm},\mathbf{CF})$.*

Proof of Theorem 1. The first inclusion—that is, $\mathscr{L}(\text{HCGR}^{(CFG_{lm},CFG)})_1 \subseteq \mathscr{L}(VAL_{lm},\mathbf{CF})$, directly follows from the fact that any time we can construct valance context-free grammar V for any $\Gamma \in$ HCGR$^{(CFG_{lm},CFG)}$ such that $L_{lm}(\Gamma)_1 = L_{lm}(V)$, where V assigns one dimension of $\mathbb{Z}^k$ and simulates computation in G_1. By valence-vector V remembers the numbers of occurrences of each nonterminal in G_2.

On the other hand, for any valence context-free grammar $V = (N_v,T_v,S_v,P_v)$, working in the leftmost way, we can construct HCGR$^{(CFG_{lm},CFG)}$ $\Gamma = (G_1,G_2,Q)$. Grammars $G_1 = (N_1,T_1,S_1,P_1)$ and $G_2 = (N_2,T_2,S_2,P_2)$ are constructed as it follows. Set $N_1 = N_v$, $T_1 = T_v$, $P_1 = \{S_1 \to S_v\Delta, \Delta \to \varepsilon\}$, $N_2 = \{S_2,A_1,A_2,\dots A_k\}$, where nonterminals $A_1,\dots,A_k$ represents components of valence vectors; $T_2 = \{\bullet\}$; set $P_2 = \{S_2 \to S_2, A_1 \to \varepsilon, A_2 \to \varepsilon,\dots,A_k \to \varepsilon, S_2 \to \varepsilon\}$; for each rule $A \to x(n_1,\dots,n_k) \in P_v$ add $r_1{:}A \to A^{(1)}$, $r_2{:}A^{(1)} \to A^{(2)}$,..., $r_{n-1}{:}A^{(n-1)} \to A^{(n)}$, $r_n{:}A^{(n)} \to x$, where $n = \Sigma_{i=1}^{max(0,-n_i)}$ ($n = 0$ implies that just $A \to x$ is added), into P_1, add $r{:}S_2 \to \widehat{x} = A_1\dots A_1A_2\dots A_2\dots A_k\dots A_kS_2$, where A_is are in $\widehat{x}$ $max(0,n_i)$-times, for all $i = 1,\ldots,k$, into P_2, add pairs of the form $(r_i, A_j \to \varepsilon)$ and (r_i, r) into Ψ in that way, where $A_j \to \varepsilon$ have to be used exactly $max(0,-n_j)$-times, for $j = 1,\ldots,k$, and (r_i, r) once during $A \Rightarrow A^{(1)} \Rightarrow \ldots \Rightarrow A^{(n)} \Rightarrow x$ derivation sequence in G_1; add $(S_1 \to S_v\Delta, S_2 \to S_2)$ and $(\Delta \to \varepsilon, S_2 \to \varepsilon)$ into Ψ.

G_1, in the constructed grammar system, step by step simulates derivation in V. Because of Ψ, whenever V makes a derivation by a rule $A \to x(n_1,\ldots,n_k)$, G_2 removes $-n_i$ A_is for all $i = 1,\ldots,k$ where $n_i < 0$ and adds n_j A_js for all $j = 1,\ldots,k$ where $n_j > 0$. As the last step of computation, G_1 and G_2 removes Δ and S_2, respectively. Formal prove that $L(\Gamma)_1 = L(V)$ is left to the reader. $\square$

In [3] has been demonstrated that leftmost valence context-free grammar is equivalent to nondeterministic valence pushdown automata. Although we do not know where exactly their class lies, it is probable that these models cannot describe all languages from **RE**.

Besides the leftmost restriction, we restrict number of nonterminals in each sentential form of $G_1,\ldots,$ G_n to one—that is, components of HCGR are linear grammars. The motivation ensues from the following theorem.

Theorem 2. *Let* $\mathscr{L}_j(HMAS^{(t_1,\ldots,t_n)}) = \{L_j\colon L_j = \{w_j\colon (w_1,\ldots,w_n) \in K\}$ *and* $K \in \mathscr{L}(HMAS^{(t_1,\ldots,t_n)})\}$, *where* $HMAS^{(t_1,\ldots,t_n)}$ *contains two* 1*-turn PDAs and* $n-2$ *FAs. Then,* $\mathscr{L}_j(HMAS^{(t_1,\ldots,t_n)}) = \mathbf{RE}$.

Proof of Lemma 2. Every recursively enumerable language can be generated by a grammar G in Geffert normal form (see [4]), i.e. by a grammar $G = (\{A,B,C,D,S\},T,S,P)$ where P contains rules only of the form $S \to uSa$, $S \to uSv$, $S \to uv$, $AB \to \varepsilon$, and $CD \to \varepsilon$, where $u \in \{A,C\}^*$, $v \in \{B,D\}^*$, $a \in T$. In addition, every sentential form of any successful derivation have to be of the form $S \Rightarrow {}^*w_1w_2w$ where $w_1w_2 \in \{A,C\}^*\{B,D\}^*$, $w \in T^*$, and $w_1w_2w \Rightarrow {}^*w$.

Let $G_1,\ldots,G_n$ be n grammars in the Geffert normal form and let $\vartheta = (M_1, M_2, \ldots, M_n, \Psi)$ be an HMAS$^{(t_1,\ldots,t_n)}$, M_1, M_2 be two 1-turn PDAs and $M_3, \ldots, M_n$ be FAs. M_1 and M_2 can generate strings over $\{A,C,|\}$ and $\{B,D,|\} \cup \{a\colon a \in \bigcup_{i=1}^{n} \Sigma_i\}$ on the M_1's and M_2's pushdown, respectively, in the following way. The PDAs start with symbol $|$ on their pushdowns. First, ϑ simulates derivations in $G_1,\ldots,G_n$. The system starts with G_1. If G_i applies a rule of the form $S_i \to uS_iw$, ϑ adds rev$u$ and w on M_1's and M_2's pushdown, respectively. If G_i make a derivation step by rule of the form $S_i \to uw$, then ϑ adds rev$(u)|$ and $w|$ on M_1's and M_2's pushdown, respectively, and the system starts with simulation of G_{i+1}. After the generation, PDAs start to compare topmost symbols on their pushdowns. At the same computation step, the automata can remove A with B and C with D. Whenever the $|$ is on the M_1's pushdown top and ϑ works on ith input string (the system starts with nth input string), M_2 compares symbols on its pushdown with input symbols which reads ith automaton. If these symbols coincide, M_2 removes the symbol from the pushdown. As $|$ is topmost pushdown symbol of both PDAs, the PDAs remove the symbol and start with $(i-1)$th input string. Clearly, *automata* in ϑ read their input only if they compare the input with content of M_2's pushdown. If all automata read all their input and the PDAs have empty pushdown, all strings are accepted—that is, ϑ accepts the input n-string.

In this way, ϑ can simulate derivations of n grammars in Geffert normal form, which generates the whole family of **RE** languages. Therefore, Lemma 2 holds. $\square$

In spite of the fact that for any language L accepted by 1-turn automata, a linear grammar generating L can be construct, Theorem 3 holds.

Theorem 3. $\mathscr{L}(HCGR^{n\,LNG}) \subsetneq \mathscr{L}(HMAS^{(t_1,\ldots,t_n)})$, *where* $HMAS^{(t_1,\ldots,t_n)}$ *has exactly two* 1*-turn PDAs and* $n-2$ *FAs.*

Proof of Theorem 3. Consider HCGR$^{n\text{LNG}}$ $\widehat{\Gamma} = (\widehat{G_1}, \dots, \widehat{G_n}, \widehat{Q})$, which generates n-language n-$L(\widehat{\Gamma})$ $= \{(w, \varepsilon, \dots, \varepsilon)\colon w \in \widehat{T_1}^*$ and $(S_1, \dots, S_n) \Rightarrow^* (w, \varepsilon, \dots, \varepsilon)\}$. Since $G_1, \dots, G_n$ are linear, any successful derivation can be expressed as $(\widehat{S_1}, \dots, \widehat{S_n}) \Rightarrow^* (uA_1v, A_2, \dots, A_n) \Rightarrow (axv, \varepsilon, \dots, \varepsilon)$, where $axv \in \widehat{T_1}^*$. The longest sentential n-form is $m + n$, where $m = |axv|$. Hence, we can construct a linear bounded automaton M with states of the form $(A_1, \dots, A_n)$, where $A_i \in \widehat{N_i} \cup \{\varepsilon\}$ and input alphabet $\Gamma = T_1 \cup \{\underline{a}\colon a \in T_1\}$. M starts with state $(\widehat{S_1}, \dots, \widehat{S_n})$ and w on its input tape. During computation, it underlines symbols on the input tape by the following algorithm. Without any loss of generality, suppose that M is in state $(A_1, \dots, A_n)$. If there are n rules, $r_1, \dots, r_n$, where r_1 is rule of the form $A \rightarrow uB_1v$, for $i = 2, \dots, n$, r_i is of the form $A_i \rightarrow B_i$ and $(r_1, \dots, r_n) \in Q$, such that u is equal to the first $|u|$ not-underlined symbols on the tape and v is equal to the last $|v|$ not-underlined symbols on the tape, M underlines these symbols and moves to the state $(B_1, \dots, B_n)$. As soon as all symbols on the tape are underlined and the automaton is in the state $(\varepsilon, \dots, \varepsilon)$, the input is accepted. Hence, $L_j = \{w\colon (w, \varepsilon, \dots, \varepsilon) \in n\text{-}L(\widehat{\Gamma})\}$ belongs to the family of **CS** languages. Therefore, Theorem 3 holds. $\square$

6 CONCLUSION

This paper continued [5] and demonstrated that generative power of the leftmost-restricted component in a n-generating grammar system with exactly one component working in the leftmost way, lies somewhere between generative power of matrix grammars and accepting power of Turing machines. The generative power of unrestricted grammars in such systems remains unsolved and it is suggested for a further research. On the other hand, the present paper showed that despite the ability of n-accepting automata system, based upon 1-turn PDAs, to accept all **RE** languages, in case of n-generating grammar system containing linear grammars, the generative power of its components lies under the accepting power of LBAs.

ACKNOWLEDGEMENTS

This work was supported by the research plan MSM0021630528, the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), and the FIT-S-11-2 BUT FIT grant.

REFERENCES

[1] B. S. Baker. Context-sesitive grammars generating context-free languages. In M. Nivat, editor, *Automata, Languages and Programming*, pages 501–506. North-Holland, Amsterdam, 1972.

[2] R. V. Book. Terminal context in context-sensitive grammars. *SIAM Journal of Computing*, 1:20–30, 1972.

[3] H. Fernau and R. Stiebe. Sequential grammars and automata with valences. *Theoretical Computer Science*, 276(1-2):377 – 405, 2002.

[4] V. Geffert. Context-free-like forms for the phrase-structure grammars. In *Proceedings of the Mathematical Foundations of Computer Science 1988*, pages 309–317, New York, 1988. Springer-Verlag.

[5] R. Lukáš and A. Meduna. Multigenerative grammar systems and matrix grammars. *Kybernetika*, 46(1):68–82, 2010.

[6] A. Meduna and R. Lukáš. Multigenerative grammar systems. *Schedae Informaticae*, 2006(15):175–188, 2006.

[7] M. Meduna. *Automata and Languages: Theory and Applications*. Springer, London, 2000.