Two-Way Metalinear PC Grammar Systems and Their Descriptional Complexity

Alexander Meduna*

Abstract

Besides a derivation step and a communication step, a two-way PC grammar system can make a reduction step during which it reduces the right-hand side of a context-free production to its left hand-side. This paper proves that every non-unary recursively enumerable language is defined by a centralized two-way grammar system, Γ, with two metalinear components in a very economical way. Indeed, Γ's master has only three nonterminals and one communication production; furthermore, it produces all sentential forms with no more than two occurrences of nonterminals. In addition, during every computation, Γ makes a single communication step. Some variants of two-way PC grammar systems are discussed in the conclusion of this paper.

1 Introduction

Over the past few years, the formal language theory has intensively investigated many variants of PC grammar systems (see [12]), which consist of several simultaneously working and communicating components, represented by grammars. This paper introduces another variant of this kind, called *two-way PC grammar systems*, which make three kinds of computational steps—derivation, reduction, and communication. More precisely, a two-way PC grammar system, Γ, makes a derivation step as usual; that is, it rewrites the left-hand side of a production with its right-hand side. During a reduction step, however, Γ rewrites the right-hand side with the left hand-side. Finally, Γ makes a communication step in a usual PC-grammar-system way; in addition, however, after making this step, it changes the computational way from derivations to reductions or vice versa.

As reduction steps represent a mathematically natural modification of derivation steps, a discussion of two-way PC grammar systems surely deserves our attention from a theoretical viewpoint. From a practical viewpoint, this discussion is important as well. Indeed, two-way PC grammar systems actually formalize computational units combining both reduction and derivation steps, which frequently occur in applied computer science. To give some specific examples, consider, for

*Department of Information Systems, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, Brno 61266, Czech Republic

instance, compilers. A parser is often written so it actually represents a combination of a bottom-up parser for expressions and a top-down parser for general program flow. While the former makes reductions, the latter makes derivations; as a whole, the parser thus makes both. To give another example in this area, the three-address code generation often consist of top-down syntax-directed generation of abstract syntax tree followed by a bottom-up translation of this tree to the desired three-address code. Again, both reductions and derivations take part in this translation process as a whole. As a result, there surely exist both theoretically and pragmatically sound reasons for investigating two-way PC grammar systems.

This paper narrows its attention to the centralized two-way metalinear PC grammar systems working in a non-returning mode. That is, since they are centralized, only their first components, called the masters, can cause these systems to make a communication step. Since they are metalinear, all their components are represented by metaliner grammars. Finally, as they work in a non-returning mode, after communicating, their components continue to process the current string rather than return to their axioms. Regarding these systems, the present paper concentrates its discussion on their descriptional complexity because this complexity represents an intensively studied area of today's formal language theory.

As its main result, this paper proves that the centralized two-way metalinear PC grammar systems characterize the family of non-unary recursively enumerable languages in a very economical way. Indeed, every non-unary recursively enumerable language is defined by a centralized two-way grammar system with two metalinear components so that during every computation Γ makes a single communication step. In addition, Γ's three-nonterminal master has only one production with a communication symbol and each of its sentential forms contains no more than two occurrences of nonterminals. In the conclusion of this paper, some terminating and parallel variants of these two-way systems are introduced and analogical results to the above characterization are achieved.

2 Preliminaries

This paper assumes that the reader is familiar with the formal language theory (see [9], [14]). For a set, Q, $card(Q)$ denotes the cardinality of Q. For an alphabet, V, V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation. For every $w \in V^*$, $|w|$ denotes the length of w. Furthermore, for every $0 \leq i \leq |w|$ and $L \in V^*$, we introduce the following denotation:

- $\mathbf{length}(L) = \{|w| : w \in L\}$
- $\mathbf{reversal}(w)$ denotes the reversal of w
- $\mathbf{reversal}(L) = \{\mathbf{reversal}(w) : w \in L\}$
- $\mathbf{alph}(w)$ denotes the set of letters occurring in w
- $\mathbf{alph}(L) = \{a : a \in \mathbf{alph}(w) \text{ with } w \in L\}$
- $\mathbf{sym}(w, i)$ denotes the ith symbol in w

- $\mathbf{prefix}(w, i)$ denotes the set of w's prefixes of length i or less
- $\mathbf{prefix}(w) = \mathbf{prefix}(w, |w|)$
- $\mathbf{suffix}(w, i)$ denotes the set of w's suffixes of length i or less
- $\mathbf{suffix}(w) = \mathbf{suffix}(w, |w|)$
- $\mathbf{prefix}(L) = \{x : x \in \mathbf{prefix}(w) \text{ for some } w \in L\}$
- $\mathbf{suffix}(L) = \{x : x \in \mathbf{suffix}(w) \text{ for some } w \in L\}$

For every $W \subseteq V$, $\mathbf{del}(w, W)$ denotes the word resulting from w by the deletion of all symbols from W in w; more formally, $\mathbf{del}(w, W) = \rho(w)$, where ρ is the weak identity over V^* defined as $\rho(b) = \varepsilon$ for every $b \in W$ and $\rho(a) = a$ for every $a \in V - W$. Let $\mathbf{keep}(w, W)$ denote the word resulting from w by the deletion of all symbols from $V - W$ in w; more formally, $\mathbf{keep}(w, W) = \theta(w)$, where θ is the weak identity over V^* defined as $\theta(b) = \varepsilon$ for every $b \in V - W$ and $\theta(a) = a$ for every $a \in W$. For instance, for $w = abac$, $\mathbf{alph}(w) = \{a, b, c\}, \mathbf{prefix}(w, 2) = \{\varepsilon, a, ab\}, \mathbf{sym}(w, 3) = a, \mathbf{del}(w, \{a\}) = bc, \mathbf{keep}(w, \{a, b\}) = aba$.

A *queue grammar* (see [7]) is a sixtuple, $Q = (V, T, W, F, s, P)$, where V and W are alphabets satisfying $V \cap W = \emptyset, T \subseteq V, F \subseteq W, s \in (V - T)(W - F)$, and $P \subseteq (V \times (W - F)) \times (V^* \times W)$ is a finite relation such that for every $a \in V$, there exists an element $(a, b, x, c) \in P$ for some $b \in W - F, x \in V^*$, and $c \in W$. If $u, v \in V^*W$ such that $u = arb, v = rzc, a \in V, r, z \in V^*, b, c \in W$, and $(a, b, x, c) \in P$, then $u \Rightarrow v$ $[(a, b, z, c)]$ in G or, simply, $u \Rightarrow v$. The language of Q, $L(Q)$, is defined as $L(Q) = \{w \in T^* : s \Rightarrow^* wf \text{ where } f \in F\}$.

Now, we slightly modify the notion of a queue grammar. A left-extended queue grammar is a sixtuple, $Q = (V, T, W, F, s, P)$, where V, T, W, F, and s have the same meaning as in a queue grammar. $P \subseteq (V \times (W - F)) \times (V^* \times W)$ is a finite relation (as opposed to an ordinary queue grammar, this definition does not require that for every $a \in V$, there exists an element $(a, b, x, c) \in P$). Furthermore, assume that $\# \notin V \cup W$. If $u, v \in V^*\{\#\}V^*W$ so that $u = w\#arb, v = wa\#rzc, a \in V, r, z, w \in V^*, b, c \in W$, and $(a, b, x, c) \in P$, then $u \Rightarrow v[(a, b, z, c)]$ in G or, simply, $u \Rightarrow v$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$; then, based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The language of Q, $L(Q)$, is defined as $L(Q) = \{v \in T^* : \#s \Rightarrow^* w\#vf$ for some $w \in V^*$ and $f \in F\}$. Less formally, during every step of a derivation, a left-extended queue grammar shifts the rewritten symbol over #; in this way, it records the derivation history, which represents a property fulfilling a crucial role in the proof of Lemma 4 in the next section.

3 Definitions

As already sketched in Section 1, this paper discusses grammar systems (see [1, 2, 3, 4, 5, 7]), concentrating its attention on PC grammar systems (see [6, 11, 12, 13, 15, 16]). The present section introduces a new version of these systems. First, based on two-way k-linear PC components, it defines two-way k-linear n-PC grammar systems. Then, it introduces several notions concerning them. Finally, two examples are given.

Let k and n be two positive integers. A two-way k-linear PC component is a quadruple, $G = (N, T, P, S)$, where N and T are two disjoint alphabets. Symbols in N and T are referred to as nonterminal and terminals, respectively, and $S \in N$ is the start symbol of G. Set $M = N - \{S\}$. P is a finite set of productions such that each $r \in P$ has one of these forms

- $S \rightarrow x$, where $x \in (T \cup M)^*$ and x contains no more than k occurrences of symbols from M,
- $A \rightarrow x$, where $A \in M$ and $x \in T^*MT^* \cup T^*$.

Let $u, v \in (N \cup T)^*$. For every $A \rightarrow x \in P$, write $uAv\ {}_d\!\Rightarrow uxv$ and $uxv\ {}_r\!\Rightarrow uAv$; d and r stand for a direct *d*erivation and a direct *r*eduction, respectively. To express that G makes $uAv\ {}_d\!\Rightarrow uxv$ according to $A \rightarrow x$, write $uAv\ {}_d\!\Rightarrow uxv\ [A \rightarrow x]$; $uxv\ {}_r\!\Rightarrow uAv\ [A \rightarrow x]$ have an analogical meaning in terms of ${}_r\!\Rightarrow$. A *two-way k-linear n-PC grammar system* is an $n+1$-tuple

$$\Gamma = (Q, G_1, \ldots, G_n),$$

where $Q = \{q_i : i = 1, \ldots, n\}$, whose members are called query symbols, and for all $i = 1, \ldots, n, G_i = (Q \cup N_i, T, P_i, S_i)$ is a two-way k-linear PC component such that $Q \cap (N_i \cup T) = \emptyset$ (notice that each G_i has the same terminal alphabet, T); let q-$P_i \subseteq P_i$ denote the set of all productions in P_i containing a query symbol. A configuration is an n-tuple of the form $(x_1, \ldots, x_n)$, where $x_i \in (Q \cup N_i \cup T)^*, 1 \leq i \leq n$. The start configuration, s, is defined as $s = (S_1, \ldots, S_n)$. Let Θ denote the set of all configurations of Γ. For every $x \in \Theta$ and $i = 1, \ldots, n, i$-x denotes its ith component—that is, if $x = (x_1, \ldots, x_i, \ldots, x_n)$, then i-$x = x_i$. For every $x \in \Theta$, define the mapping ${}_x\theta$ over $\{i\text{-}x : 1 \leq i \leq n\}$ as ${}_x\theta(i\text{-}x) = z_1 z_2 \ldots z_{|i\text{-}x|}$ where for all $1 \leq h \leq |i\text{-}x|$,

if for some $q_j \in Q, i = 1, \ldots, n, \mathbf{sym}(i\text{-}x, h) = q_j$ and $\mathbf{alph}(j\text{-}x) \cap Q = \emptyset$, then $z_h = j\text{-}x$; otherwise (that is, $\mathbf{sym}(i\text{-}x, h) \notin Q$ or $\mathbf{alph}(j\text{-}x) \cap Q \neq \emptyset), z_h = \mathbf{sym}(i\text{-}x, h)$.

Let $y, x \in \Theta$. Write

- $y\ {}_d\!\Rightarrow x$ in Γ if i-$y\ {}_d\!\Rightarrow i$-x in G_i or i-$y = i$-x with i-y, i-$x \in T^*$, for all $i = 1, \ldots, n$;
- $y\ {}_r\!\Rightarrow x$ in Γ if i-$y\ {}_r\!\Rightarrow i$-x in G_i or i-$y = i$-x with i-y, i-$x \in \{S_i\} \cup T^*$, for all $i = 1, \ldots, n$;
- $y\ {}_q\!\Rightarrow x$ in Γ if i-$x = {}_y\theta(i\text{-}y)$ in G_i for all $i = 1, \ldots, n$.

Informally, Γ works in three computational modes—${}_d\!\Rightarrow, {}_r\!\Rightarrow, {}_q\!\Rightarrow$, which symbolically represent a direct *derivation*, *reduction*, and *communication*, respectively. Let $l \geq 1, \alpha_j \in \Theta, 1 \leq i \leq l$, and $\alpha_0\ {}_{l_1}\!\Rightarrow \alpha_1\ {}_{l_2}\!\Rightarrow \alpha_2 \ldots \alpha_{l-1}\ {}_{l_l}\!\Rightarrow \alpha_l$ where $l_m \in \{d, r, q\}, 1 \leq m \leq l$; write $\alpha_0 \Rightarrow^* \alpha_l$ if $l_1 = d$ and each $l_p \in \{d, r, q\}, 2 \leq p \leq l-1$, satisfies:

- if $l_p = q$ then $l_{p+1}, l_{p-1} \in \{d, r\}$ and $l_{p+1} \neq l_{p-1}$,
- if $l_p \in \{d, r\}$ then $l_{p+1} \in \{q, l_p\}$.

Informally, after making a communication step, Γ changes the computational mode from d to r and vice versa; after making a derivation or reduction step, it does not. Consider $\alpha_0 \Rightarrow^* \alpha_l$ that consists of l direct computational steps, $\alpha_0 \; {}_{l_1}\!\Rightarrow \alpha_1 \; {}_{l_2}\!\Rightarrow \alpha_2 \dots \alpha_{l-1} \; {}_{l_l}\!\Rightarrow \alpha_l$, satisfying the above properties. Set $\kappa(\alpha_0 \Rightarrow^* \alpha_l) = \{\alpha_0, \alpha_1, \dots, \alpha_l\}$; that is, $\kappa(\alpha_0 \Rightarrow^* \alpha_l)$ denote the set of all configurations occurring in $\alpha_0 \Rightarrow^* \alpha_l$. Furthermore, for each $l = 1, \dots, n$, set $\kappa(i\text{-}\alpha_0 \Rightarrow^* i\text{-}\alpha_l) = \{i\text{-}\beta : \beta \in \kappa(\alpha_0 \Rightarrow^* \alpha_l)\}$. Finally, for each $h = 1, \dots, n$, *h-computation*$(i\text{-}\alpha_0 \Rightarrow^* i\text{-}\alpha_l)$ denote $h\text{-}\alpha_0 \; {}_{l_1}\!\Rightarrow h\text{-}\alpha_1 \; {}_{l_2}\!\Rightarrow h\text{-}\alpha_2 \dots h\text{-}\alpha_{l-1} \; {}_{l_l}\!\Rightarrow h\text{-}\alpha_l$ The *language* of Γ, $L(\Gamma)$, is defined as

$$L(\Gamma) = \{z \in T^* : \sigma \Rightarrow^* \alpha \text{ in } \Gamma \text{ with } z = \mathbf{del}(1\text{-}a, S_1), \text{ for some } \alpha \in \Theta\}.$$

Informally, $L(\Gamma)$ contains $z \in T^*$ if and only if there exists $\alpha \in \Theta$ such that $\sigma \Rightarrow^* a$ in Γ and the deletion of each S_1 in $1\text{-}a$ results in z. A computation $\sigma \Rightarrow^* a$ in Γ with $\mathbf{del}(1\text{-}a, S_1) \in L(\Gamma)$ is said to be successful. By a *two-way metalinear n-PC grammar system*, we refer to any two-way k-linear n-PC grammar system, where $k \geq 1$.

Notice that after communicating, the components of the above systems continue to process the current string rather than return to their axioms. In other words, they work in the non-returning mode (see [7]). The returning mode is not discussed in this paper.

For a two-way k-linear PC grammar system, $\Gamma = (Q, G_1, \dots, G_n)$, we next introduce some special notions.

Finite index. Let $\sigma \Rightarrow^* x$ be any successful computation in Γ, where $x \in \Theta$, and let $i \in \{1, \dots, n\}$. By *i-index*$(\sigma \Rightarrow^* x)$, we denote the maximum number in $\mathbf{length}(\mathbf{keep}(\kappa(i\text{-}\sigma \Rightarrow^* i\text{-}x), N_i))$. If for every successful computation $\sigma \Rightarrow^* \xi$ in Γ, where $\xi \in \Theta$, there exists $k \geq 1$ such that *i-index*$(\sigma \Rightarrow^* \xi) \leq k$, G_i is of a *finite index*. If G_i is of a finite index, *index*(G_i) denotes the minimum number h satisfying *i-index*$(\sigma \Rightarrow^* \xi) \leq h$, for every successful computation $\sigma \Rightarrow^* \varpi$ in Γ, where $\varpi \in \Theta$. By *index*$(G_i) = \infty$, we express that G_i is not of a finite index. If G_j is of a finite index for all $j = 1, \dots, n$, Γ is of a *finite index* and *index*(Γ) denotes the minimum number g satisfying *index*$(G_l) \leq g$, for all $l = 1, \dots, n$. By *index*$(\Gamma) = \infty$, we express that Γ is not of a finite index.

q-Degree. For $\sigma \Rightarrow^* x$ in Γ, where $x \in \Theta$, *q-degree*$(\sigma \Rightarrow^* x)$ denotes the number of communication steps $({}_q\!\Rightarrow)$ in $\sigma \Rightarrow^* x$. If for every computation $\sigma \Rightarrow^* \xi$ in Γ, where $\xi \in \Theta$, there exists $k \geq 1$ such that *q-degree*$(\sigma \Rightarrow^* \xi) \leq k$, Γ is of a *finite q-degree*. If Γ is of a finite q-degree, *q-degree*(Γ) denotes the minimum number h satisfying *q-degree*$(\sigma \Rightarrow^* \xi) \leq h$, for every computation $\sigma \Rightarrow^* \xi$ in Γ; by *q-degree*$(\Gamma) = \infty$, we express that Γ is not of a finite q-degree.

Centralized Version. Γ is *centralized* if no query symbol occurs in any production of P_i in $G_i = (N_i, T_i, P_i, S_i)$, for all $i = 2, \dots, n$. In other words, only P_1 can contain some query symbols, so G_1, called the *master* of Γ, is the only component that can cause Γ to perform a communication step.

This paper concentrates its attention on the centralized version of two-way k-linear 2-PC grammar systems. Therefore, we conclude this section by two examples illustrating these systems.

Example 1. Consider the centralized two-way two-linear 2-PC grammar system, $G = (\{q_1, q_2\}, G_1, G_2)$, where $G_1 = (\{S_1, A, B\}, T, P_1, S_1), G_2 = (\{S_2, B, Y\}, T, P_2, S_2), T = \{a, b, c\}$, $P_1 = \{S_1 \to A, A \to cA, A \to cq_2, Q_2 \to B, B \to q_2, B \to \varepsilon, S_1 \to B\}$, and $P_2 = \{S_2 \to YB, B \to B, Y \to aYb, Y \to ab\}$.

For instance, Γ generates $c^3a^3b^3a^3b^3a^3b^3$ as $(S_1, S_2) \;_d{\Rightarrow}\; (A, YB) \;_d{\Rightarrow}\; (cA, aYbB) \;_d{\Rightarrow}\; (ccA, aaYbbB) \;_d{\Rightarrow}\; (cccq_2, a^3b^3B) \;_q{\Rightarrow}\; (c^3a^3b^3B, a^3b^3B) \;_r{\Rightarrow}\; (c^3a^3b^3q_2, a^3b^3B) \;_q{\Rightarrow}\; (c^3a^3b^3a^3b^3B, a^3b^3B) \;_d{\Rightarrow}\; (c^3a^3b^3a^3b^3q_2, a^3b^3B) \;_q{\Rightarrow}\; (c^3a^3b^3a^3b^3a^3b^3B, a^3b^3B) \;_r{\Rightarrow}\; (c^3a^3b^3a^3b^3a^3b^3S_1, a^3b^3B)$ with $\mathbf{del}(c^3a^3b^3a^3b^3a^3b^3S_1, S_1) = c^3a^3b^3a^3b^3a^3b^3$.

Observe that $L(\Gamma) = \{c^jx^i : x \in H, j, i \geq 1, |x| = 2j\}$, where $H = \{a^nb^n : n \geq 1\}$. Furthermore, notice that $index(G_1) = 1$ and $index(G_2) = 2$, so Γ is of a finite index. On the other hand, $q\text{-}degree(\Gamma) = \infty$.

Example 2. Consider the centralized two-way one-linear 2-PC grammar system $G = (\{q_1, q_2\}, G_1, G_2)$ where $G_1 = (\{S_1, A, B\}, T, P_1, S_1), G_2 = (\{S_2, B\}, T, P_2, S_2), T = \{a, b, c\}, P_1 = \{S_1 \to A, A \to aAa, A \to aq_2a, B \to Bc, S_1 \to B\}$, and $P_2 = \{S_2 \to B, B \to bBc\}$.

For instance, Γ makes $(S_1, S_2) \;_d{\Rightarrow}\; (A, B) \;_d{\Rightarrow}\; (aAa, bBc) \;_d{\Rightarrow}\; (aaq_2aa, bbBcc) \;_q{\Rightarrow}\; (aabbBccaa, bbBcc) \;_r{\Rightarrow}\; (aabbBcaa, bBc) \;_r{\Rightarrow}\; (aabbS_1caa, B)$.

Notice that $L(\Gamma) = \{a^nb^nc^ma^n : n \geq m \geq 0\}, index(G_1) = 1, index(G_2) = 1$, and $q\text{-}degree(\Gamma) = 1$.

4 Main Result

This section proves that every non-unary recursively enumerable language is defined by a centralized two-way three-linear 2-PC grammar system, $\Gamma = (\{Q_2\}, G_1, G_2)$, such that $index(G_1) = 2, index(G_2) = 3$, and $q\text{-}degree(\Gamma) = 1$. As a result, $index(\Gamma) = 3$. In addition, its three-nonterminal master, G_1, has only one production containing a query symbol.

Lemma 1. *For every recursively enumerable language, L, there exists a left-extended queue grammar, Q, satisfying $L(Q) = L$.*

Proof. Recall that every recursively enumerable language is generated by a queue grammar (see [8]). Clearly, for every queue grammar, there exists an equivalent left-extended queue grammar. Thus, this lemma holds. ■

Lemma 2. *Let Q' be a left-extended queue grammar. Then, there exists a left-extended queue grammar, $Q = (V, T, W, F, s, R)$, such that $L(Q') = L(Q)$, $W = X \cup Y \cup \{1\}$, where $X, Y, \{1\}$ are pairwise disjoint, and every $(a, b, x, c) \in R$ satisfies either $a \in V - T, b \in X, x \in (V - T)^*, c \in X \cup \{1\}$ or $a \in V - T, b \in Y \cup 1, x \in T^*, c \in Y$.*

Proof. See Lemma 1 in [10]. ■

Consider the left-extended queue grammar $Q = (V, T, W, F, s, R)$ from Lemma 2. Its properties imply that Q generates every word in $L(Q)$ so that it passes through state 1. Before it enters 1, it generates only words over $(V - T)$; after entering 1, it generates only words over T. In greater detail, the next corollary expresses this property, which fulfills a crucial role in the proof of Lemma 4.

Corollary 3. *Q constructed in the proof of Lemma 2 generates every $h \in L(Q)$ in this way*

$$\begin{array}{lll}
 & \#a_0q_0 & \\
\Rightarrow & a_0\#x_0q_1 & [(a_0, q_0, z_0, q_1)] \\
\Rightarrow & a_0a_1\#x_1q_2 & [(a_1, q_1, z_1, q_2)] \\
\vdots & & \\
\Rightarrow & a_0a_1 \dots a_k\#x_kq_{k+1} & [(a_k, q_k, z_k, q_{k+1})] \\
\Rightarrow & a_0a_1 \dots a_ka_{k+1}\#x_{k+1}y_1q_{k+2} & [(a_{k+1}, q_{k+1}, y_1, q_{k+2})] \\
\vdots & & \\
\Rightarrow & a_0a_1 \dots a_ka_{k+1} \dots a_{k+m-1} & \\
 & \#x_{k+m-1}y_1 \dots y_{m-1}q_{k+m} & [(a_{k+m-1}, q_{k+m-1}, y_{m-1}, q_{k+m})] \\
\Rightarrow & a_0a_1 \dots a_ka_{k+1} \dots a_{k+m}\#y_1 \dots y_mq_{k+m+1} & [(a_{k+m}, q_{k+m}, y_m, q_{k+m+1})]
\end{array}$$

where $k, m \geq 1, a_i \in V - T$ for $i = 0, \dots, k + m, x_j \in (V - T)^*$ for $j = 1, \dots, k + m, s = a_0q_0, a_jx_j = x_{j-1}z_j$ for $j = 1, \dots, k, a_1 \dots a_kx_{k+1} = z_0 \dots z_k, a_{k+1} \dots a_{k+m} = x_k, q_0, q_1, \dots, q_{k+m} \in W - F$ and $q_{k+m+1} \in F, z_1, \dots, z_k \in (V - T)^*, y_1, \dots, y_m \in T^*, h = y_1y_2 \dots y_{m-1}y_m$.

Lemma 4. *Let Q be a left-extended queue grammar such that card$(\mathbf{alph}(L(Q))) \geq 2$. Then, there exists a centralized two-way three-linear 2-PC grammar system, $\Gamma = (\{Q_2\}, G_1, G_2)$, such that $L(\Gamma) = L(Q)$, index$(G_1) = 2$, index$(G_2) = 3$, index$(\Gamma) = 3$, q-degree$(\Gamma) = 1$. In addition, Γ's master, $G_1 = (\{Q_2\} \cup N_1, T, P_1, S_1)$, satisfies card$(N_1) = 3$ and q-$P_1 = \{A \to Q_2\}$.*

Proof. Let $Q = (V, T, W, F, s, R)$ be a left-extended queue grammar such that *card*$(\mathbf{alph}(L(Q))) \geq 2$. Assume that $\{0, 1\} \subseteq \mathbf{alph}(L(\Gamma)) \cap T$. Furthermore, without any loss of generality, assume that Q satisfies the properties described in Lemma 2 and Corollary 3. Observe that there exist a positive integer, n, and an injection, ι, from VW to $(\{0, 1\}^n - 1^n)$ so that ι remains an injection when its domain is extended to $(VW)^*$ in the standard way (after this extension, ι thus represents an injection from $(VW)^*$ to $(\{0, 1\}^n - 1^n)^*$); a proof of this observation is simple and left to the reader. Based on ι, define the substitution, ν, from V to $(\{0, 1\}^n - 1^n)$ as $\nu(a) = \{\iota(aq) : q \in W\}$ for every $a \in V$. Extend the domain of ν to V^*. Furthermore, define the substitution, μ, from W to $(\{0, 1\}^n - 1^n)$ as $\mu(q) = \{\mathbf{reversal}(\iota(aq)) : a \in V\}$ for every $q \in W$. Extend the domain of μ to W^*. Set $o = 1^n$.

Construction. Introduce the centralized two-way three-linear 2-PC grammar system, $\Gamma = (\{Q_2\}, G_1, G_2)$, where $G_1 = (Q \cup N_1, T, P_1, S_1), G_2 = (N_2, T, P_2, S_2)$, $N_1 = \{S_1, A, Y\}$, and $P_1 = \{S_1 \rightarrow oAo, S_1 \rightarrow oYo, A \rightarrow Q_2\} \cup \{A \rightarrow \mathbf{reversal}(x)Ax : x \in \iota(VW)\} \cup \{Y \rightarrow xYx : x \in \iota(VW)\}$. P_2 is constructed as follows

1. if $s = a_0q_0$, where $a_0 \in V - T$ and $q_0 \in W - F$, then add $S_2 \rightarrow Yu\,\langle q_0, 1\rangle\, tY$ to P_2, for all $u \in \nu(a_0)$ and $t \in \mu(q_0)$,
2. if $(a, q, y, p) \in R$, where $a \in V - T, p, q \in W - F$, and $y \in (V - T)^*$, then add $\langle q, 1\rangle \rightarrow u\,\langle p, 1\rangle\, t$ to P_2, for all $u \in \nu(y)$ and $t \in \mu(p)$,
3. for every $q \in W - F$, add $\langle q, 1\rangle \rightarrow o\,\langle q, 2\rangle$ to P_2,
4. if $(a, q, y, p) \in R$, where $a \in V - T, p, q \in W - F, y \in T^*$, then add $\langle q, 2\rangle \rightarrow y\,\langle p, 2\rangle\, t$ to P_2, for all $t \in \mu(p)$,
5. if $(a, q, y, p) \in R$, where $a \in V - T, q \in W - F, y \in T^*$, and $p \in F$, then add $\langle q, 2\rangle \rightarrow yo$ to P_2,
6. add $Y \rightarrow Y$ to P_2,

and N_2 contains all symbols occurring in P_2 that are not in T.

Basic Idea. Clearly, Γ's master, $G_1 = (\{Q_2\} \cup N_1, T, P_1, S_1)$, satisfies $card(N_1) = 3$ and q-$P_1 = \{A \rightarrow Q_2\}$. Every generation of $y \in L(\Gamma)$ can be expressed as follows

$$\begin{array}{ll}
 & (S_1, S_2) \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_0)A\beta_0 o, Y\chi_0\,\langle q_1, 1\rangle\,\mathbf{reversal}(\beta_0)Y) \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_1)A\beta_1 o, Y\chi_1\,\langle q_2, 1\rangle\,\mathbf{reversal}(\beta_1)Y) \\
\vdots & \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_k)A\beta_k o, Y\chi_k\,\langle q_{k+1}, 1\rangle\,\mathbf{reversal}(\beta_k)Y) \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_k)A\beta_k o, Y\chi_k o\,\langle q_{k+1}, 2\rangle\,\mathbf{reversal}(\beta_k)Y) \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_k)A\beta_{k+1} o, Y\chi_k o y_1\,\langle q_{k+1}, 2\rangle\,\mathbf{reversal}(\beta_{k+1})Y) \\
\vdots & \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_{k+m})Q_2\beta_{k+m} o, Y\chi_k o y_1 \ldots y_m o\mathbf{reversal}(\beta_{k+m})Y) \\
{}_q\Rightarrow & (o\mathbf{reversal}(\alpha_{k+m})Y\alpha_{k+m} o y_1 \ldots y_m o\mathbf{reversal}(\beta_{k+m})Y\beta_{k+m} o), \zeta) \\
{}_r\Rightarrow & (o\mathbf{prefix}(\mathbf{reversal}(\alpha_{k+m}), |\alpha_{k+m}| - n)Y\mathbf{suffix}(a_{k+m}, |a_{k+m}| - n) \\
 & oy_1 \ldots y_m o\mathbf{reversal}(\beta_{k+m})Y\beta_{k+m} o), \zeta) \\
\vdots & \\
{}_r\Rightarrow & (oYoy_1 \ldots y_m oYo, \zeta) \\
{}_r\Rightarrow^2 & (S_1 y_1 \ldots y_m S_1, \zeta)
\end{array}$$

where $k, m \geq 1$, and for all $e = 0, \ldots, k + m, \alpha_e \in \nu(a_0 \ldots a_e), \beta_e \in \mu(q_0 \ldots q_e)$, $\alpha_e = \mathbf{reversal}(\beta_e), a_i \in V - T, q_i \in W - F, 1 \leq i \leq k + m$, for all $f = 0, \ldots, k - 1$, $\chi_f \in \mathbf{prefix}(\nu(a_0 \ldots a_e)) \cap \mathbf{prefix}(\chi_{f+1}), \chi_k = a_{k+m}, s = a_0q_0, y_1, \ldots, y_m \in T^*$,

$\zeta = Y\chi_k o y_1 \ldots y_m o\mathbf{reversal}(\beta_{k+m})Y, y = y_1, \ldots, y_m$, and R contains rules $(a_0, q_0, z_0, q_1), (a_1, q_1, z_1, q_2), \ldots, (a_{k+m}, q_{k+m}, y_{m-1}, q_{k+m+1})$ according to which Q can make the generation of y described in Corollary 3. As a result, q-*degree*$(\Gamma) = 1$ and $L(\Gamma) \subseteq L(Q)$. On the other hand, recall that Q generates every $y \in L(Q)$ as described in Corollary 3. Then, we can easily construct the above generation of y in Γ, so $L(Q) \subseteq L(\Gamma)$. Therefore, $L(\Gamma) = L(Q)$.

Formal Proof (Sketch). For brevity, the following rigorous proof omits some obvious details, which the reader can easily fill in.

Claim 1. *G generates every $h \in L(\Gamma)$ as follows $(S_1, S_2)\ {}_d\Rightarrow^* (uAv, y)\ {}_q\Rightarrow (uyv, y)\ {}_r\Rightarrow^* (h, y)$, where $u, v \in \{0,1\}^*, y \in \{Y\}(T \cup \{0,1\})^*\{Y\}$.*

Proof. In P_1, the right-hand side of every production contains a symbol from $Q \cup N_1$, so during any successful computation, Γ makes at least one q-step. The only production by which G_1 can cause Γ to make a q-step is $A \to q_2$. A does not occurr in N_2 at all, and after the first application of $A \to q_2$, G_1 makes reductions during which it can never obtain A in a sentential form. Thus, the first application of $A \to q_2$ is also the last application of this production. Therefore, Γ generates every $h \in L(\Gamma)$ as follows $(S_1, S_2)\ {}_d\Rightarrow^* (uAv, y)\ {}_q\Rightarrow (uyv, y)\ {}_r\Rightarrow^* (h, z)$,where $u, v \in \{0,1\}^*, y, z \in (T \cup N)^*$. If y contains a symbol from $N_2 - (T \cup \{Y\})$, G_1 can never remove them during $(uyv, y)\ {}_r\Rightarrow^* (h, z)$ by any rule from P_1, which leads to a contradiction that $h \neq L(\Gamma)$. Thus, $y, z \in (T \cup \{Y\})^*$. Examine P_2 to see that $y, z \in (T \cup \{Y\})^*$ implies $y = z$ and $y \in \{Y\}(T \cup \{0,1\})^*\{Y\}$. As a result, $(S_1, S_2)\ {}_d\Rightarrow^* (uAv, y)\ {}_q\Rightarrow (uyv, y)\ {}_r\Rightarrow^* (h, y)$, where $u, v \in \{0,1\}^*, y \in \{Y\}(T \cup \{0,1\})^*\{Y\}$. □

The previous claim implies q-*degree*$(\Gamma) = 1$.

Claim 2. *Let $(S_1, S_2)\ {}_d\Rightarrow^* (uAv, y)\ {}_q\Rightarrow (uyv, y)\ {}_r\Rightarrow^* (h, y)$ in Γ, where $h \in L(\Gamma), u, v \in \{0,1\}^*, y \in \{Y\}(T \cup \{0,1\})^*\{Y\}$. Then, $v = \mathbf{reversal}(u)$.*

Proof. Examine 1-P_1. Observe that before the communicational step, G_1 can use only productions from $\{S_1 \to oAo\} \cup \{A \to \mathbf{reversal}(z)Az : z \in \iota(VW)\}$; therefore, $v = \mathbf{reversal}(u)$. □

Claim 3. *Let $(S_1, S_2)\ {}_d\Rightarrow^* (uA\mathbf{reversal}(u), y)\ {}_q\Rightarrow (uy\mathbf{reversal}(u), y)\ {}_r\Rightarrow^* (h, y)$, in Γ, where $h \in L(\Gamma), u, v \in \{0,1\}^*, y \in \{Y\}(T \cup \{0,1\})^*\{Y\}$. Then, $y = Y\mathbf{reversal}(u)huY$.*

Proof. Consider $(uy\mathbf{reversal}(u), y)\ {}_r\Rightarrow^* (h, y)$. During 1-*computation*$((uy\ \mathbf{reversal}(u), y)\ {}_r\Rightarrow^* (h, y))$, G_1 can use only productions from $\{S_1 \to oYo\} \cup \{Y \to xYx : x \in \iota(VW)\}$. Thus, $y = Y\mathbf{reversal}(u)huY$. □

Return to the proof of the lemma. Let

$$\begin{aligned}(S_1, S_2)\ &{}_d\Rightarrow^* && (uA\mathbf{reversal}(u), Y\mathbf{reversal}(u)huY)\\ &{}_q\Rightarrow && (uY\mathbf{reversal}(u)huY\mathbf{reversal}(u), Y\mathbf{reversal}(u)huY)\\ &{}_r\Rightarrow^* && (h, Y\mathbf{reversal}(u)huY)\end{aligned}$$

in Γ, where $u, v \in \{0,1\}^*$. Examine P_1 and P_2 to see that in greater detail this computation can be expressed as

$$\begin{array}{ll}
& (S_1, S_2) \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_0)A\beta_0 o, Y\chi_0 \langle q_1, 1\rangle \mathbf{reversal}(\beta_0)Y) \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_1)A\beta_1 o, Y\chi_1 \langle q_2, 1\rangle \mathbf{reversal}(\beta_1)Y) \\
\vdots & \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_k)A\beta_k o, Y\chi_k \langle q_{k+1}, 1\rangle \mathbf{reversal}(\beta_k)Y) \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_k)A\beta_k o, Y\chi_k o \langle q_{k+1}, 2\rangle \mathbf{reversal}(\beta_k)Y) \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_k)A\beta_{k+1} o, Y\chi_k o y_1 \langle q_{k+1}, 2\rangle \mathbf{reversal}(\beta_{k+1})Y) \\
\vdots & \\
{}_d\Rightarrow & (o\mathbf{reversal}(\alpha_{k+m})Q_2\beta_{k+m} o, Y\chi_k o y_1 \dots y_m o\mathbf{reversal}(\beta_{k+m})Y) \\
{}_q\Rightarrow & (o\mathbf{reversal}(\alpha_{k+m})Y\alpha_{k+m} o y_1 \dots y_m o\mathbf{reversal}(\beta_{k+m})Y\beta_{k+m}o), \zeta) \\
{}_r\Rightarrow & (o\mathbf{prefix}(\mathbf{reversal}(\alpha_{k+m}), |\alpha_{k+m}| - n)Y\mathbf{suffix}(a_{k+m}, |a_{k+m}| - n) \\
& o y_1 \dots y_m o\mathbf{reversal}(\beta_{k+m})Y\beta_{k+m}o), \zeta) \\
\vdots & \\
{}_r\Rightarrow & (oYoy_1 \dots y_m oYo, \zeta) \\
{}_r\Rightarrow^2 & (S_1 y_1 \dots y_m S_1, \zeta)
\end{array}$$

where $k, m \geq 1$, and for all $e = 0, \dots, k+m, \alpha_e \in \nu(a_0 \dots \alpha_e), \beta_e \in \mu(q_0 \dots q_e)$, $\alpha_e = \mathbf{reversal}(\beta_e), a_i \in V - T, q_i \in W - F, 1 \leq i \leq k+m$, for all $f = 0, \dots, k-1, \chi_f \in \mathbf{prefix}(\nu(a_0 \dots a_e)) \cap \mathbf{prefix}(\chi_{f+1}), \chi_k = \alpha_{k+m}, s = a_0 q_0$, $y_1, \dots, y_m \in T^*, \zeta = Y\chi_k o y_1 \dots y_m o\mathbf{reversal}(\beta_{k+m})Y, h = y_1, \dots, y_m$. Thus, $index(G_1) = 2, index(G_2) = 3$, and $index(\Gamma) = 3$. Recall that $\chi_k = a_{k+m}$. Consider the derivation part of the above computation—that is,

$$2\text{-}computation((S_1, S_2)\ {}_d\Rightarrow^* \quad (o\mathbf{reversal}(\alpha_{k+m})Q_2\beta_{k+m}o, Y\alpha_{k+m}oy_1 \dots y_m o\mathbf{reversal}(b_{k+m})Y))$$

From the construction of P_2, the form of this computation implies that R contains rules $(a_0, q_0, z_0, q_1), (a_1, q_1, z_1, q_2), \dots, (a_{k+m}, q_{k+m}, y_{m-1}, q_{k+m+1})$, where $s = a_0 q_0, a_j x_j = x_{j-1} z_j$ for $j = 1, \dots, k, a_1 \dots a_k x_{k+1} = z_0 \dots z_k, a_{k+1} \dots a_{k+m} = x_k$, and $q_{k+m+1} \in F, z_1, \dots, z_k \in (V-T)^*, y_1, \dots, y_m \in T^*, h = y_1 y_2 \dots y_{m-1} y_m$. As a result,

$$\begin{array}{lll}
& \#a_0 q_0 & \\
\Rightarrow & a_0 \# x_0 q_1 & [(a_0, q_0, z_0, q_1)] \\
\Rightarrow & a_0 a_1 \# x_1 q_2 & [(a_1, q_1, z_1, q_2)] \\
\vdots & & \\
\Rightarrow & a_0 a_1 \dots a_k \# x_k q_{k+1} & [(a_k, q_k, z_k, q_{k+1})] \\
\Rightarrow & a_0 a_1 \dots a_k a_{k+1} \# x_{k+1} y_1 q_{k+2} & [(a_{k+1}, q_{k+1}, y_1, q_{k+2})] \\
\vdots & & \\
\Rightarrow & a_0 a_1 \dots a_k a_{k+1} a_{k+m-1} \# x_{k+m-1} y_1 \cdots y_{m-1} q_{k+m} & \\
& & [(a_{k+m-1}, q_{k+m-1}, y_{m-1}, q_{k+m})] \\
\Rightarrow & a_0 a_1 \dots a_k a_{k+1} a_{k+m} \# y_1 \dots y_m q_{k+m+1} & [(a_{k+m}, q_{k+m}, y_m, q_{k+m+1})]
\end{array}$$

in Q. As $h = y_1y_2 \dots y_{m-1}y_m, h \in L(Q)$. Thus, $L(\Gamma) \subseteq L(Q)$.

To prove $L(Q) \subseteq L(\Gamma)$, recall that Q satisfies the properties described in Lemma 2 and, therefore, generates every $h \in L(Q)$ as described in Corollary 3. Then, we can easily construct the generation of h in Γ that has the form described above; a detailed version of this construction is left to the reader. Thus, $h \in L(\Gamma)$, so $L(Q) \subseteq L(\Gamma)$.

Therefore, $L(\Gamma) = L(Q)$. Recall that we have already established that $index(G_1) = 2, index(G_2) = 3, index(\Gamma) = 3, q - degree(\Gamma) = 1, card(N_1) = 3$, $q\text{-}P_1 = \{AQ_2\}$. Thus, Lemma 4 holds. ■

Theorem 5. *Let L be a recursively enumerable language such that $card(\mathbf{alph}(L)) \geq 2$. Then, there exists a centralized two-way three-linear 2-PC grammar system, $\Gamma = (\{q_2\}, G_1, G_2)$, such that $L(\Gamma) = L, index(G_1) = 2, index(G_2) = 3, index(\Gamma) = 3, q\text{-}degree(\Gamma) = 1$, and Γ's master, $G_1 = (Q \cup N_1, T, P_1, S_1)$, satisfies $card(N_1) = 3, q\text{-}P_1 = \{A \to Q_2\}$.*

Proof. This theorem follows from Lemmas 1, 2, and 4. ■

5 Some Variants

This concluding section discusses some variants of the centralized two-way metalinear grammar systems.

Parallel variant. A parallel variant of a centralized two-way k-linear PC grammar system makes communication steps as defined in Section 4; however, during derivation and reduction steps, it allows their components to simultaneously rewrite the word at several places. More formally, let $\Gamma = (Q, G_1, \dots, G_n)$, where for all $i = 1, \dots, n, G_i = (Q \cup N_i, T, P_i, S_i)$ is a two-way k-linear PC component. As before, for $u, v \in (N_i \cup T)^*$ and $A \to x \in P_i$, write $uAv \,{}_d\!\Rightarrow uxv$ and $uxv \Rightarrow ruAv$ in G_i. Let $x_i, y_i \in (N \cup T)^*$, where $i = 1, \dots, n$, for some $n \geq 1$. If $x_i \,{}_d\!\Rightarrow y_i$ in G_i for all $i = 1, \dots, n$, write $x_1 \dots x_n \,{}_{par\text{-}d}\!\Rightarrow y_1 \dots y_n$ in Γ. If $x_i \,{}_r\!\Rightarrow y_i$ in G_i for all $i = 1, \dots, n$, write $x_1 \dots x_n \,{}_{par\text{-}r}\!\Rightarrow y_1 \dots y_n$ in Γ. To complete the definition of a parallel centralized two-way k-linear PC grammar system, modify the corresponding definition given in Section 3 by substituting ${}_{par\text{-}d}\!\Rightarrow$ and ${}_{par\text{-}r}\!\Rightarrow$ for ${}_d\!\Rightarrow$ and ${}_r\!\Rightarrow$, respectively. By ${}_{par}L(\Gamma)$, denote the language generated by a parallel two-way k-linear PC grammar system, Γ.

Theorem 6. *Let L be a recursively enumerable language such that $card(\mathbf{alph}(L)) \geq 2$. Then, there exists a parallel centralized two-way three-linear 2-PC grammar system, $\Gamma = (\{Q_2\}, G_1, G_2)$, such that ${}_{par}L(\Gamma) = L, index(G_1) = 2, index(G_2) = 3, index(\Gamma) = 3, q\text{-}degree(\Gamma) = 1$, and Γ's master, $G_1 = (Q \cup N_1, T, P_1, S_1)$, satisfies $card(N_1) = 3$ and $q\text{-}P_1 = \{A \to Q_2\}$.*

Proof. Establish this theorem by analogy with the demonstration of Theorem 5. ■

Terminating mode. The theory of grammar systems has introduced several derivation modes, such as *-mode or the maximal code for CD grammar systems, and studied the corresponding families of languages generated in these modes. In terms of the grammar systems discussed in this paper, we also suggest a new derivation mode, called the *terminating mode.* That is, for a centralized 2-PC two-way metalinear grammar system, Γ, introduced in Section 3, the *language generated by* Γ *in the terminating mode,* ${}_tL(\Gamma)$, is defined by this equivalence: $L(\Gamma)$ contains $z \in T^*$ if and only if there exists $\alpha \in \Theta$ such that Γ makes $\sigma \Rightarrow^* \alpha$ but cannot make any further computational step from α and the deletion of each S_1 in 1-α results in z.

Theorem 7. *Let L be a recursively enumerable language such that $card(\mathbf{alph}(L))) \geq 2$. Then, there exists a parallel centralized two-way three-linear 2-PC grammar system, $\Gamma = (\{Q_2\}, G_1, G_2)$, such that ${}_tL(\Gamma) = L, index(G_1) = 2, index(G_2) = 3, index(\Gamma) = 3, q\text{-}degree(\Gamma) = 1$, and Γ's master, $G_1 = (Q \cup N_1, T, P_1, S_1)$, satisfies $card(N_1) = 4$ and $q\text{-}P_1 = \{A \rightarrow Q_2\}$.*

Proof. Return to the centralized two-way metalinear 2-PC grammar system, $\Gamma = (\{Q_2\}, G_1, G_2)$, constructed in the proof of Lemma 4. Modify its master, $G_1 = (Q \cup N_1, T, P_1, S_1)$, as follows. First, add a new nonterminal, X, to N_1. Then, include $\{X \rightarrow X\} \cup \{X \rightarrow xYy \mid x, y \in \iota(VW), x \neq y\}$ into P_1. Complete this proof by analogy with the proofs of Lemma 4 and Theorem 5. ■

Returning mode. As stated in Section 1, this paper considers only the non-returning mode throughout. Reconsider the present study in terms of returning mode (see [7]).

Acknowledgement

The author thanks the anonymous referee for several useful comments. The author also gratefully acknowledge support of GAČR grant 201/04/0441.

References

[1] Csuhaj-Varju, E.: Cooperating Grammar Systems. Power and Parameters, *LNCS* 812, Springer, Berlin, 67-84, 1994.

[2] Csuhaj-Varju, E.: Grammar Systems: a Multi-Agent Framework for Natural Language Generation, in Gh. Paun (ed.), *Artificial Life: Grammatical Models*, The Black Sea Univer. Press, Bucharest, 1995.

[3] Csuhaj-Varju, E. and Kelemen, J.: On the Power of Cooperation: a Regular Representation of R.E. Languages, *Theor. Computer Sci.* 81, 305-310, 1991.

[4] Csuhaj-Varju, E., Dassow, J., Kelemen, J., Paun, Gh.: G*rammar Systems: A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.

[5] Csuhaj-Varju, E., Dassow, J., Kelemen, J., Paun, Gh.: Eco-Grammar Systems: A Grammatical Framework for Life-like Interactions, *Artificial Life* 3, 27-38, 1996.

[6] Csuhaj-Varju, E. and Salomaa, A.: Networks of Language Processors: Parallel Communicating Systems, *EATCS Bulletin* 66, 122-138, 1997.

[7] Dassow, J., Paun, Gh., and Rozenberg, G.: Grammar Systems. In *Handbook of Formal Languages*, Rozenberg, G. and Salomaa, A. (eds.), Volumes 2, Springer, Berlin 1997

[8] Kleijn, H. C. M. and Rozenberg, G.: On the Generative Power of Regular Pattern Grammars, *Acta Informatica* 20, 391-411, 1983.

[9] Meduna, A.: *Automata and Languages: Theory and Applications*, Springer, London, 2000.

[10] Meduna, A.: Simultaneously One-Turn Two-Pushdown Automata, *International Journal of Computer Mathematics* 80, 679-687, 2003.

[11] Paun, Gh., Salomaa, A. and S. Vicolov, S.: On the Generative Capacity of Parallel Communicating Grammar Systems, *International Journal of Computer Mathematics* 45, 45-59, 1992.

[12] Paun, Gh. and Santean, L.: Parallel Communicating Grammar Systems: the Regular Case, *Ann. Univ. Buc., Ser. Matem.–Inform.* 38, 55-63, 1989.

[13] Paun, Gh. and Santean, L.: Further Remarks about Parallel Communicating Grammar Systems, *International Journal of Computer Mathematics* 34, 187-203, 1990.

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

[15] Santean, L.: Parallel Communicating Systems, *EATCS Bulletin*, 160-171, 1990.

[16] Vaszil, G.: On simulating Non-returning PC grammar Systems with Returning Systems, *Theoretical Computer Science* (209) 1-2, 319-329, 1998.

Received February, 2003