H. Leung and G. Pighizzini (Eds.): DCFS 2006, Las Cruces, NM, USA, Proceedings.
Computer Science Technical Report NMSU-CS-2006-001, pp. 241-249, 2006.

On the Degree Complexity of Special Non-Context-Free Languages with Respect to PC Grammar Systems*

Jürgen Dassow and Bianca Truthe†
Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik
Postfach 4120, D-39016 Magdeburg, Germany
{dassow,truthe}@iws.cs.uni-magdeburg.de

Abstract

When modeling natural languages, three languages are of special interest. In natural languages, there occur phenomena like multiple agreements, crossed agreements and replication. These aspects are represented by the three languages $K_1 = \{a^n b^n c^n \mid n \geq 1\}$, $K_2 = \{a^n b^m c^n d^m \mid m, n \geq 1\}$ and $K_3 = \{ww \mid w \in \{a, b\}^+\}$, respectively.

In the present paper, we give parallel communicating grammar systems (PC grammar systems) that generate the languages K_1, K_2 and K_3 but use less or less powerful components than those systems published so far. This improves existing results.

1 Introduction and Definitions

A parallel communicating grammar system (PC grammar system) consists of several grammars with their own production rules and sentential forms. For solving a task, the components (grammars) work simultaneously and are allowed to communicate. According to the monograph [2], a communication is done by request: a component can request the whole word generated by another component. A minimal synchronization is assumed: In each time unit every component carries out a rewriting step or the system performs a communication step. If no component wants to communicate, then each grammar derives its current sentential form by a rewriting step according to its production rules (if a grammar has reached already a terminal word, it keeps it and does not disturb the other components; if the sentential form of a grammar contains only those nonterminals for which the grammar has no rules, then the component blocks the whole system). If components communicate, then all requests are satisfied simultaneously (if possible). If a component which has been asked has sent a request to another component, then it first waits for an answer and second it sends its own answer. In this situation, it can happen that two components wait for each other – this leads to a deadlock of the system. In the returning mode, each component restarts with its start symbol after satisfying a request; in the non-returning mode the components continue with their current words.

*The authors would like to thank György Vaszil and the anonymous referees for their remarks.
†Corresponding author

Those components that are not involved in a communication do nothing in this moment; especially, they do not perform a rewriting step. If a component does not get its request satisfied, its sentential form remains unchanged.

We give the formal definitions now.

Let $n \geq 1$ be a natural number. A PC grammar system of degree n is an $(n+3)$-tuple

$$\Gamma = (N, T, K, (P_1, S_1), \ldots, (P_n, S_n))$$

where N is a finite set (the set of nonterminals), T is a finite set (the set of terminals) with $T \cap N = \emptyset$, $K = \{ Q_1, \ldots, Q_n \}$ is a set of n request symbols ($K \cap (N \cup T) = \emptyset$) and (P_i, S_i) are the components of the system, where $S_i \in N$ and $G_i = (N \cup K, T, P_i, S_i)$ is a Chomsky grammar ($i = 1, \ldots, n$).

A PC grammar system is called centralized, if request symbols are introduced only by the first component, and non-centralized otherwise. A PC grammar system is called regular, right-linear, linear, context-free etc., if the rules in the components have the respective property. In [6], it was shown that centralized PC grammar systems with right-linear components are more powerful than centralized systems with regular components. A rule $A \to B$ without a terminal is called a chain rule.

A configuration of a PC grammar system with n components is an n-tuple $(w_1, \ldots, w_n)$ with $w_i \in (N \cup T \cup K)^*$. A PC grammar system $\Gamma = (N, T, K, (P_1, S_1), \ldots, (P_n, S_n))$ derives a configuration $(x_1, \ldots, x_n)$ to a configuration $(y_1, \ldots, y_n)$ in the returning or non-returning mode (written formally as $(x_1, \ldots, x_n) \Longrightarrow (y_1, \ldots, y_n)$ maybe with the index R for 'returning' or N for 'non-returning'), if one of the following cases holds:

1. No word x_i, $i = 1, \ldots, n$, contains a request symbol.
   For $i = 1, \ldots, n$, we have $x_i \Longrightarrow_{G_i} y_i$ or $x_i \in T^*$ and $y_i = x_i$.

2. A word x_i, $i = 1, \ldots, n$, contains a request symbol.
   For each index $i = 1, \ldots, n$ we have: If the word x_i contains a request symbol, then there are a natural number $m_i \geq 1$, words $z_{i,1}, \ldots, z_{i,m_i+1} \in (N \cup T)^*$ and natural numbers $t_{i,1}, \ldots, t_{i,m_i} \in \{ 1, \ldots, n \}$ such that $x_i = z_{i,1} Q_{t_{i,1}} \cdots z_{i,m_i} Q_{t_{i,m_i}} z_{i,m_i+1}$. If a word $x_{t_{i,j}}$ ($j \in \{ 1, \ldots, m_i \}$) contains a request symbol, then $y_i = x_i$ otherwise $y_i = z_{i,1} x_{t_{i,1}} z_{i,2} x_{t_{i,2}} \cdots z_{i,t_i} x_{t_{i,m_i}} z_{i,m_i+1}$. If the word x_i does not contain a request symbol, we have $y_i = x_i$ in the non-returning mode and there are two cases in the returning mode:

   - There is a word x_k with the request symbol Q_i and such that $|x_{t_{k,j}}|_K = 0$ for all $j = 1, \ldots, m_k$. Then $y_i = S_i$ (if the i-th component is asked but does not ask itself, it returns to the start symbol after answering).
   - There is no such word x_k. Then we have $y_i = x_i$ (if the i-th component is not involved in communication, the sentential form does not change).

By $\Longrightarrow^*$ we denote the reflexive and transitive closure of the relation $\Longrightarrow$.

The configuration $(S_1, \ldots, S_n)$ is called the start configuration; each configuration $(w_1, \ldots, w_n)$ with $w_1 \in T^*$ is called an end configuration of the system Γ, if it can be reached from the start configuration, i. e. if $(S_1, \ldots, S_n) \Longrightarrow^* (w_1, \ldots, w_n)$.

The languages $L_{\mathsf{R}}(\Gamma)$ and $L_{\mathsf{N}}(\Gamma)$ generated by a PC grammar system Γ in the returning and non-returning mode, respectively, are the sets of all words $w \in T^*$, for which words $w_2, \ldots, w_n$ exist such that $(w, w_2, \ldots, w_n)$ is an end configuration of Γ:

$$L_x(\Gamma) = \{\, w \in T^* \mid (S_1, \ldots, S_n) \Longrightarrow_x^* (w, w_2, \ldots, w_n) \,\} \text{ for } x \in \{\, \mathsf{R}, \mathsf{N} \,\}.$$

For $Y \in \{\, PC, CPC \,\}$ and $X \in \{\, REG, RL, LIN, CF \,\}$, we denote by

- PC_nX the set of all PC grammar systems (centralized or not) with at most n components of type X,
- CPC_nX the set of all centralized PC grammar systems with at most n components of type X,
- $\mathcal{L}_{\mathsf{R}}(Y_nX)$ the set of all languages that are generated by a Y_nX-system in the returning mode,
- $\mathcal{L}_{\mathsf{N}}(Y_nX)$ the set of all languages that are generated by a Y_nX-system in the non-returning mode,
- $\mathcal{L}_{\mathsf{R+N}}(Y_nX) = \mathcal{L}_{\mathsf{R}}(Y_nX) \cap \mathcal{L}_{\mathsf{N}}(Y_nX)$ the set of all languages that are generated by a Y_nX-system in the returning mode and by a (possibly different) Y_nX-system in the non-returning mode, and
- $\mathcal{L}(Y_nX) = \mathcal{L}_{\mathsf{R}}(Y_nX) \cup \mathcal{L}_{\mathsf{N}}(Y_nX)$ the set of all languages that are generated by a Y_nX-system in any of the modes.

By $\#_a(w)$ we quote the number of occurrences of the letter a in the word w.

By cooperation, even regular grammars are able to generate non-context-free languages (examples can be found in [2] and [4]). This is important since context-free languages are not sufficient for the modeling of natural languages and also artificial languages (a more detailed discussion can be read in [3]). Three non-context-free phenomena occurring with natural languages are multiple agreements, crossed agreements and replication. They are represented by the languages $K_1 = \{\, a^n b^n c^n \mid n \geq 1 \,\}$, $K_2 = \{\, a^n b^m c^n d^m \mid m \geq 1,\ n \geq 1 \,\}$ and $K_3 = \left\{\, ww \;\middle|\; w \in \{\, a, b \,\}^+ \,\right\}$. In the following sections, we give some PC grammar systems for generating these languages and discuss their tightness. By reasons of space we give only sketches of proofs; for more detailed proofs we refer to [5].

2 Special PC Grammar Systems

Now we study the languages K_1, K_2 and K_3 introduced above.

2.1 PC Grammar Systems for the Languages K_1 and K_3

In [4], $K_1 \in \mathcal{L}_{\mathsf{R+N}}(CPC_3RL)$ was shown. We show that $K_1 \in \mathcal{L}_{\mathsf{R+N}}(CPC_3REG)$ holds. Let $\Gamma_1 = (\{S_1, S_1', S_2, S_2', S_3, S_3', B, C\}, \{a, b, c\}, \{Q_1, Q_2, Q_3\}, (P_1, S_1), (P_2, S_2), (P_3, S_3))$ with

$P_1 = \{S_1 \to aB, B \to bC, C \to c, S_1 \to aS_1', S_1' \to aS_1', S_1' \to aQ_2, S_1 \to aQ_2, S_2' \to bQ_3, S_3' \to c\}$,
$P_2 = \{S_2 \to aS_2', S_2' \to bS_2'\}$ and $P_3 = \{S_3 \to bS_3', S_3' \to cS_3'\}$.

At the beginning, the first component has three rules: $S_1 \to aB$, $S_1 \to aS_1'$ and $S_1 \to aQ_2$. In the first case, we obtain the derivation

$$(S_1, S_2, S_3) \Longrightarrow (aB, aS_2', bS_3') \Longrightarrow (abC, abS_2', bcS_3') \Longrightarrow (abc, abbS_2', bccS_3').$$

The second case leads, after n more steps, to the configuration $(a^{n+1}Q_2, ab^nS_2', bc^nS_3')$. Hence, the second and third case can be considered together:

$$\begin{aligned}(S_1, S_2, S_3) &\Longrightarrow^{n\geq 1} (a^nQ_2, ab^{n-1}S_2', bc^{n-1}S_3') \Longrightarrow (a^{n+1}b^{n-1}S_2', *, bc^{n-1}S_3') \\ &\Longrightarrow (a^{n+1}b^nQ_3, *, bc^nS_3') \Longrightarrow (a^{n+1}b^{n+1}c^nS_3', *, *) \Longrightarrow (a^{n+1}b^{n+1}c^{n+1}, *, *).\end{aligned}$$

If a component is not asked any more and does not block the system, then the component does not affect the computation any longer. In this case the component is marked by a star $*$. The PC grammar system Γ_1 generates the language K_1 in both modes. The rules of the system are regular (and not only right-linear).

Proposition 1 *We have* $L_{\mathsf{R}}(\Gamma_1) = L_{\mathsf{N}}(\Gamma_1) = K_1$ *and* $K_1 \in \mathcal{L}_{\mathsf{R+N}}(CPC_3REG)$.

According to [2, Theorem 7.9] the language K_1 does not belong to the class $\mathcal{L}_{\mathsf{R}}(CPC_2RL)$ and hence not to the class $\mathcal{L}_{\mathsf{R}}(CPC_2REG)$, since it is not context-free. The following theorem states that the language K_1 can not be generated by a CPC_2RL-system in the non-returning mode, either.

Theorem 2 $K_1 \notin \mathcal{L}_{\mathsf{N}}(CPC_2RL)$.

Proof. We assume $K_1 = L_{\mathsf{N}}(\Gamma)$ for a centralized parallel communicating grammar system $\Gamma = (N, \{a, b, c\}, \{Q_1, Q_2\}, (P_1, S_1), (P_2, S_2))$ with two components in the non-returning mode. Let $k_1 = \#(N) + 1$, $k_2 = \max\{|x| | A \to xB \in P_1 \cup P_2,\ x \in \{a, b, c\}^*,\ B \in N \cup \{\lambda\}\}$ and $k = (k_1 + 2) \cdot k_2$. We now consider the word $z = a^{2k}b^{2k}c^{2k} \in L_{\mathsf{N}}(\Gamma)$. We distinguish three cases.

Case 1. The word z is generated without communication steps.

By using standard techniques and a pumping lemma, we obtain that $L_{\mathsf{N}}(\Gamma)$ contains a word $a^{2k}b^{2k}c^{2k+s}$ with $s \geq 1$ which contradicts $K_1 = L_{\mathsf{N}}(\Gamma)$.

Case 2. The word z is generated with exactly one communication step.

Here we have one of the following derivations

$$(S_1, S_2) \Longrightarrow^* (\alpha_1Q_2, \beta_1B) \Longrightarrow (\alpha_1\beta_1B, \beta_1B) \Longrightarrow^* (\alpha_1\beta_1\alpha_2, \beta_1\beta_2B') \tag{1}$$

with derivations $D_1 = S_1 \Longrightarrow^* \alpha_1Q_2$ and $D_2 = B \Longrightarrow^* \alpha_2$ with respect to the first component, a derivation $D_3 = S_2 \Longrightarrow^* \beta_1B$ with respect to P_2, $\alpha_1\beta_1\alpha_2 = a^{2k}b^{2k}c^{2k}$ and D_1 and D_3 have the same length, or

$$(S_1, S_2) \Longrightarrow^* (\alpha_1Q_2, \beta_1B) \Longrightarrow (\alpha_1\beta_1B, \beta_1B) \Longrightarrow^* (\alpha_1\beta_1\alpha_2, \beta_1\beta_2) \quad \text{or} \tag{2}$$

$$(S_1, S_2) \Longrightarrow^* (\alpha_1Q_2, \beta_1) \Longrightarrow (\alpha_1\beta_1, \beta_1). \tag{3}$$

We discuss only (1); the argumentation for (2) and (3) works analogously.

We consider the following two subcases.

Case 2.1. The word α_1 contains the letter c.

Then $\alpha_1 = a^{2k}b^{2k}c^r$ for some $r > 0$ and $\beta_1 = c^s$ for some s. Then we have a derivation

$$\begin{aligned} D = S_1 &\Longrightarrow^* a^{t_0}B_0 \Longrightarrow^* a^{t_0}a^{t_1}B_1 \Longrightarrow^* a^{t_0}a^{t_1}a^{t_2}B_2 \\ &\Longrightarrow^* a^{t_0}a^{t_1}a^{t_2}a^{t_3}B_3 \Longrightarrow^* \cdots \Longrightarrow^* a^{t_0}a^{t_1}\cdots a^{t_{k_1}}B_{k_1} \Longrightarrow^* a^{2k}b^{2k}c^rQ_2 \end{aligned}$$

with respect to P_1, where $t_i > 0$. By the choice of k_1 there are numbers p and q such that $p < q$ and $B_p = B_q$. With respect to P_2, we have a derivation

$$E = S_2 \Longrightarrow^* c^{l_0}C_0 \Longrightarrow^* c^{l_0}c^{l_1}C_1 \Longrightarrow^* \cdots \Longrightarrow^* c^{l_0}c^{l_1}\cdots c^{l_{k_1}}C_{k_1} \Longrightarrow^* c^sB$$

such that the lengths of the derivations $S_1 \Longrightarrow^* a^{t_0}\cdots a^{t_i}B_i$ and $S_2 \Longrightarrow^* c^{l_0}\cdots c^{l_i}C_i$ are equal. Obviously, there are numbers p' and q' such that $p' < q'$ and $C_{p'} = C_{q'}$. Let u_1 and u_2 be the length of the subderivations $B_p \Longrightarrow^* a^tB_q$ and $C_{p'} \Longrightarrow^* c^lC_{q'}$, respectively. We now construct the derivations D' and E' from D and E by performing u_2-times and u_1-times the subderivations $B_p \Longrightarrow^* a^tB_q$ and $C_{p'} \Longrightarrow^* c^lC_{q'}$ in addition, respectively (i. e., in both derivations we perform $u_1 \cdot u_2$ additional derivation steps). This leads to the derivations

$$\begin{aligned} (S_1, S_2) &\Longrightarrow^* (a^{2k+u_2t}b^{2k}c^rQ_2, c^{s+u_1l}B) \Longrightarrow (a^{2k+u_2t}b^{2k}c^rc^{s+u_1l}B, c^{s+u_1l}B) \\ &\Longrightarrow^* (a^{2k+u_2t}b^{2k}c^{2k+u_1l}, c^{s+u_1l}\beta_2). \end{aligned}$$

This implies $a^{2k+u_2t}b^{2k}c^{2k+u_1l} \in L_{\mathsf{N}}(\Gamma)$. Since $a^{2k+u_2t}b^{2k}c^{2k+u_1l} \notin K_1$, we have a contradiction.

Case 2.2. The word α_1 does not contain the letter c.

A contradiction to $K_1 = L_{\mathsf{N}}(\Gamma)$ is obtained similarly to Case 2.1.

Case 3. The word z is generated with at least two communication steps.

Now we have a derivation

$$\begin{aligned} (S_1, S_2) &\Longrightarrow^* (\alpha_1Q_2, \beta_1B_1) \Longrightarrow (\alpha_1\beta_1B_1, \beta_1B_1) \\ &\Longrightarrow^* (\alpha_1\beta_1\alpha_2Q_2, \beta_1\beta_2B_2) \Longrightarrow (\alpha_1\beta_1\alpha_2\beta_1\beta_2B_2, \beta_1\beta_2B_2) \\ &\Longrightarrow^* (\alpha_1\beta_1\alpha_2\beta_1\beta_2\cdots\beta_{r-1}\alpha_r\beta_1\beta_2\cdots\beta_r\alpha_{r+1}, \beta_1\beta_2\beta_3\cdots\beta_r\beta_{r+1}B_{r+1}) \\ &= (z, \beta_1\beta_2\beta_3\cdots\beta_r\beta_{r+1}B_{r+1}) \end{aligned}$$

or a derivation of the analogous type but a terminating derivation $S_2 \Longrightarrow^* \beta_1\beta_2\cdots\beta_i$, $1 \leq i \leq r+1$ (which corresponds to $\beta_j = \lambda$ for $i+1 \leq j \leq r+1$). We discuss only the above presented derivation.

Since β_1 occurs at least two times as a subword of z, β_1 cannot contain occurrences of two different letters from $\{a, b, c\}$. Thus $\beta_1 = x^r$ for some $x \in \{a, b, c\}$ and some $r \geq 0$. Moreover, $\beta_2, \beta_3, \ldots, \beta_r$ are also words in $\{ x \}^*$.

We only discuss $x = c$. The argumentation for $x = a$ and $x = b$ can be given by some modification. If $x = c$, then we obtain a contradiction by consideration analogous to those in Subcase 2.1 since we can pump as in α_1 without a change of the number of bs.

Consequently in all cases we obtain a contradiction which shows that our assumption is false. Thus $K_1 \notin \mathcal{L}_{\mathsf{N}}(CPC_2RL)$. □

The language K_1 cannot be generated by a CPC_2RL-system (regardless of the mode). Hence the result above (Proposition 1) is optimal with respect to the type of the rules.

If we admit linear components, the number of the components can be reduced to two. Let $\Gamma_2 = (\{ S_1, S_2, S_1' \}, \{ a, b, c \}, \{ Q_1, Q_2 \}, (P_1, S_1), (P_2, S_2))$ with

$P_1 = \{ S_1 \to aS_1'c,\ S_1' \to aS_1'c,\ S_1' \to a^2Q_2c^2,\ S_1 \to a^2Q_2c^2,\ S_2 \to b,\ S_1 \to abc \}$ and
$P_2 = \{ S_2 \to bS_2 \}$.

If the first component does not derive the terminal word abc, then we obtain the derivation $(S_1, S_2) \Longrightarrow^n (a^{n+1}Q_2c^{n+1}, b^nS_2) \Longrightarrow (a^{n+1}b^nS_2c^{n+1}, *) \Longrightarrow (a^{n+1}b^{n+1}c^{n+1}, *)$ with $n \geq 1$. The system Γ_2 generates in both modes the language K_1.

Theorem 3 $K_1 \in \mathcal{L}_{\mathsf{R+N}}(CPC_3REG) \cap \mathcal{L}_{\mathsf{R+N}}(CPC_2LIN) \setminus \mathcal{L}(CPC_2RL)$.

We now consider $K_3 = \{ ww \mid w \in \{ a, b \}^+ \}$. In [4] and [1], $K_3 \in \mathcal{L}_{\mathsf{R+N}}(CPC_2CF)$ and $K_3 \in \mathcal{L}_{\mathsf{N}}(CPC_2RL)$ were proved, respectively. According to [2, Theorem 7.10], we have $K_3 \notin \mathcal{L}_{\mathsf{R}}(PC_2RL)$. However, three components are sufficient.

Theorem 4 $K_3 \in \mathcal{L}_{\mathsf{R}}(PC_3RL)$.

Proof. Let $\Gamma_3 = (N, \{ a, b \}, \{ Q_1, Q_2, Q_3 \}, (P_1, S_1), (P_2, S_2), (P_3, S_3))$ with

$N = \{ S_1, S_2, S_3, A, B, \bar{A}, \bar{B}, C_a, C_b, E_a, E_b \}$,
$P_1 = \{ S_1 \to A,\ S_1 \to B,\ S_1 \to \bar{A},\ S_1 \to \bar{B},\ S_1 \to Q_2,\ C_a \to aQ_3,\ C_b \to bQ_3 \}$,
$P_2 = \{ S_2 \to Q_1,\ A \to aQ_1,\ B \to bQ_1,\ \bar{A} \to C_a,\ \bar{B} \to C_b,\ E_a \to a,\ E_b \to b \}$,
$P_3 = \{ S_3 \to Q_1,\ A \to aQ_1,\ B \to bQ_1,\ \bar{A} \to E_a,\ \bar{B} \to E_b,\ E_a \to E_a,\ E_b \to E_b \}$.

The first component decides which letter has to be produced next, the second and the third component follow the instructions from the first one such that both components produce the same word. In the end (marked by $\bar{A}$ or $\bar{B}$), the second component carries the information for the first component that there is still something to get, whereas the third component has the information that the word is finished. □

2.2 PC Grammar Systems for the Language K_2

In [4], a context-free PC grammar system with three components and a context-free PC grammar system with ten components were given that generate the language K_2 in the returning and non-returning mode, respectively. In [1], a context-free, centralized PC grammar system with four components was given that generates this language in the returning mode, and one with five components that generates it in the non-returning mode. In this section, we give linear and right-linear PC grammar systems with less components and hence improve the existing results.

2.2.1 Linear Systems

The language K_2 is generated in both modes by a linear, centralized PC grammar system with three components and in each mode also by a linear PC grammar system with two components.

Theorem 5 $K_2 \in \mathcal{L}_{\mathsf{R+N}}(CPC_3LIN) \cap \mathcal{L}_{\mathsf{R+N}}(PC_2LIN)$.

Proof. We first give a CPC_3LIN-system which generates the language K_2 in both derivation modes. Let $\Gamma_4 = (N, \{ a, b, c, d \}, \{ Q_1, Q_2, Q_3 \}, (P_1, S_1), (P_2, S_2), (P_3, S_3))$ with

$$
\begin{aligned}
N &= \{ S_1, S_2, S_3, A, B, C, D, A', B', C', D', \bar{B}, \bar{C} \}, \\
P_1 &= \{ S_1 \to D',\ D' \to Dd,\ D \to D',\ D \to aQ_2cd^2,\ B \to Q_3,\ \bar{B} \to b,\ S_1 \to A',\ A' \to aA \} \\
&\quad \cup \{ A \to A',\ A' \to a^2bQ_2d,\ C \to Q_3,\ \bar{C} \to c,\ S_1 \to abcd,\ S_1 \to aQ_2cd^2 \}, \\
P_2 &= \{ S_2 \to B, S_2 \to aBc, B \to B', B' \to B, B' \to aBc, S_2 \to C', C' \to C, C' \to bCd, C \to C' \}, \\
P_3 &= \{ S_3 \to bB',\ B' \to \bar{B},\ \bar{B} \to bB',\ S_3 \to \bar{C},\ \bar{C} \to cC',\ C' \to \bar{C} \}.
\end{aligned}
$$

The first component (called the master) derives the word $abcd$ directly or decides for generating the ds or as. If the master generates ds, then it will ask the second component (assistent) in an odd numbered step and receives a 'useful' answer (with the nonterminal B) only if the assistent has generated the as and cs, otherwise the answer contains a nonterminal (namely C') that the master cannot derive. Later the master asks the third assistent in an even numbered rewriting step and receives a useful answer only if it generated the bs. Hence, the only successful derivation in this case is

$$
\begin{aligned}
(S_1, S_2, S_3) &\Longrightarrow^{1+2k} (aQ_2cd^{k+2}, a^iBc^i, b^{k+1}B') && k \geq 0,\ 0 \leq i \leq k+1 \\
&\Longrightarrow (a^{i+1}Bc^{i+1}d^{k+2}, *, b^{k+1}B') && \Longrightarrow (a^{i+1}Q_3c^{i+1}d^{k+2}, *, b^{k+1}\bar{B}) \\
&\Longrightarrow (a^{i+1}b^{k+1}\bar{B}c^{i+1}d^{k+2}, *, *) && \Longrightarrow (a^{i+1}b^{k+2}c^{i+1}d^{k+2}, *, *).
\end{aligned}
$$

In this case, Γ_4 generates the language $L_1 = \{ a^nb^mc^nd^m \mid m \geq 2,\ 1 \leq n \leq m \}$.

If the master generates as, then it will ask the second assistent in an even numbered step and the third assistent in an odd numbered rewriting step. In this case, the PC grammar system Γ_4 generates the language $L_2 = \{ a^nb^mc^nd^m \mid n \geq 2,\ 1 \leq m \leq n \}$.

Each assistent will be asked by the master exactly once and will never stop working. Therefore it makes no difference whether an assistent continues or returns after being asked. Hence, the linear, centralized PC grammar system Γ_4 generates the language $\{ abcd \} \cup L_1 \cup L_2 = K_2$ in the returning and non-returning mode. This proves the part $K_2 \in \mathcal{L}_{\mathsf{R+N}}(CPC_3LIN)$ of the assertion.

Now we will give a PC_2LIN-system that generates the language K_2 in the returning mode. Let $\Gamma_5 = (\{ S_1, S_2, B, D, T \}, \{ a, b, c, d \}, \{ Q_1, Q_2 \}, (P_1, S_1), (P_2, S_2))$ with

$$
\begin{aligned}
P_1 &= \{ S_1 \to T,\ T \to T,\ S_1 \to Q_2d,\ S_1 \to Dd,\ D \to Dd,\ D \to Q_2d,\ B \to b \}, \\
P_2 &= \{ S_2 \to aS_2c,\ S_2 \to aQ_1c,\ T \to B,\ B \to bB \}.
\end{aligned}
$$

The second component first generates the as and cs. Afterwards it asks the master for the 'agreement' to start producing bs by receiving and processing the nonterminal T. This works only if the master does not produce ds too early. If the second component generates bs, but the first one does not produce ds after returning, then the system will never end. Otherwise the first component generates the ds and asks sometime the second component for the rest of the word.

For the non-returning mode, a similar system can be constructed. Since the second component is asked only once, it is unimportant whether it works returning or non-returning. This component remains unchanged. For working correctly in the non-returning mode, the first component has to be in a well defined state when the second component is asking. This state has to ensure that on the one hand the second component can start producing bs and on the other hand the first component starts generating ds. From these considerations we obtain $\Gamma_6 = (\{ S_1, S_2, B, D, T \}, \{ a, b, c, d \}, \{ Q_1, Q_2 \}, (P_1, S_1), (P_2, S_2))$

with

$P_1 = \{\, S_1 \to S_1,\ S_1 \to T,\ T \to Q_2 d,\ T \to Dd,\ D \to Dd,\ D \to Q_2 d,\ B \to b \,\}$ and
$P_2 = \{\, S_2 \to aS_2c,\ S_2 \to aQ_1c,\ T \to B,\ B \to bB \,\}$.

In both systems Γ_5 and Γ_6, the nonterminal T is the only nonterminal that the second component can process. Hence, the following derivation is the only one which leads to a terminal word:

$$\begin{aligned}(S_1, S_2) &\Longrightarrow^{n\geq 1} (T, a^n Q_1 c^n) \Longrightarrow (w_1, a^n T c^n) \Longrightarrow^{m\geq 1} (Q_2 d^m, a^n b^{m-1} B c^n) \\ &\Longrightarrow (a^n b^{m-1} B c^n d^m, w_2) \Longrightarrow (a^n b^m c^n d^m, w_2'),\end{aligned}$$

where we have in the returning mode $w_1 = S_1$ and $w_2 = S_2$, in the non-returning mode $w_1 = T$ and $w_2 = a^n b^{m-1} B c^n$ and in both cases $w_2 \Longrightarrow w_2'$.

The PC grammar systems Γ_5 and Γ_6 prove the part $K_2 \in \mathcal{L}_{\mathsf{R+N}}(PC_2LIN)$ of the assertion. □

2.2.2 Right-linear Systems

In [7], $K_2 \notin \mathcal{L}(PC_2RL)$ and $K_2 \in \mathcal{L}_{\mathsf{N}}(PC_3RL)$ were proved.

Theorem 6 $K_2 \in \mathcal{L}_{\mathsf{R+N}}(CPC_4RL) \cap \mathcal{L}_{\mathsf{R}}(PC_3RL)$.

Proof. Let $\Gamma_7 = (N, \{\, a, b, c, d \,\}, \{\, Q_1, Q_2, Q_3, Q_4 \,\}, (R_1, S_1), (R_2, S_2), (R_3, S_3), (R_4, S_4))$ with

$$\begin{aligned}N =& \{S_1, S_2, S_3, A_1', A_2', A_3', B_1', B_2', B_3', A_1, \ldots, A_{11}, B_1, B_2, B_3, C_2, C_3, C_4, T_2, T_3, T_4, Z_1, \ldots, Z_{10}\},\\
R_1 =& \{S_1 \to A_1', A_1' \to A_2', A_2' \to Q_3, A_2' \to aA_3', A_3' \to A_1', S_1 \to B_1', B_1' \to B_2', B_2' \to Q_2\}\\
&\cup \{B_2' \to B_3', B_3' \to B_1', T_3 \to Z_1, Z_1 \to Z_2, Z_2 \to Q_4, T_4 \to aZ_3, Z_3 \to Q_3, Z_3 \to Z_4\}\\
&\cup \{Z_4 \to T_4, T_2 \to aQ_3, T_2 \to aZ_5, Z_5 \to Z_6, Z_6 \to T_2, C_3 \to Z_7, Z_7 \to Z_8, Z_8 \to Q_2\}\\
&\cup \{C_2 \to Z_9, Z_9 \to Z_{10}, Z_{10} \to Q_4, C_4 \to d\},\\
R_2 =& \{S_2 \to A_1,\ A_1 \to A_2,\ A_2 \to A_3,\ \ldots,\ A_8 \to A_9,\ A_9 \to cA_{10},\ A_{10} \to C_2,\ C_2 \to A_9\}\\
&\cup \{S_2 \to B_1,\ B_1 \to B_2,\ B_2 \to B_3,\ B_3 \to B_1,\ B_2 \to T_2,\ T_2 \to A_{11},\ A_{11} \to A_9\},\\
R_3 =& \{S_3 \to A_1,\ A_1 \to A_2,\ A_2 \to A_3,\ A_3 \to A_1,\ A_2 \to T_3,\ T_3 \to A_4,\ A_4 \to A_5\}\\
&\cup \{A_5 \to A_6,\ A_6 \to bA_7,\ A_7 \to C_3,\ C_3 \to A_6,\ S_3 \to B_1,\ B_1 \to A_6\},\\
R_4 =& \{S_4 \to A_1,\ A_1 \to A_2,\ A_2 \to A_3,\ A_3 \to A_1,\ A_2 \to T_4,\ T_4 \to A_4,\ A_4 \to A_5\}\\
&\cup \{A_5 \to A_6,\ \ldots,\ A_9 \to A_{10},\ A_{10} \to C_4,\ C_4 \to dA_9, S_4 \to B_1,\ B_1 \to A_{11},\ A_{11} \to A_4\}.\end{aligned}$$

There are two suitable successors of the start configuration only: (A_1', A_1, A_1, A_1) and (B_1', B_1, B_1, B_1). In the first case (the A-way), all words $a^n b^m c^n d^m$ with $m \geq 1$ and $n \geq m$ are generated. The first component produces a^{n-m} first, then it asks the third component whether it is ready to start with the bs, if so then it asks the fourth component for the agreement to start with the ds. Then the first component produces the remaining a^m, receives the bs, cs and finally the ds.

In the second case (the B-way), all words $a^n b^m c^n d^m$ with $n \geq 1$ and $m \geq n$ are generated. The first component waits until the third component has produced b^{m-n}, then it asks the second one whether it is ready to start with the cs. If so, then each component generates the remaining n letters of each type.

In all other cases, the system blocks, because the first component receives a nonterminal which cannot be derived. Hence, Γ_7 produces in the non-returning mode the language K_2.

For a CPC_4RL-system in the returning mode, one simply has to replace the rules $T_2 \to A_{11}$ by $S_2 \to A_{11}$, $T_3 \to A_4$ by $S_3 \to A_4$ and $T_4 \to A_4$ by $S_4 \to A_4$.

The idea behind the PC_3RL-system given in [7] for the non-returning mode can be realized as a PC_3RL-system for the returning mode. A system is given here, the reader may convince himself of its working and correctness:

$$\Gamma_8 = (N, \{ a, b, c, d \}, \{ Q_1, Q_2, Q_3 \}, (P_1, S_1), (P_2, S_2), (P_3, S_3))$$

with

$$N = \{S_1, S_2, S_3, S_1', S_2', S_3', A, B, C, D, C', C'', X\},$$
$$P_1 = \{S_1 \to S_1', S_1' \to A, A \to aA, S_1 \to Q_2, S_1 \to X, X \to X, X \to Q_2, B \to Q_3, D \to d\},$$
$$P_2 = \{S_2 \to S_2', S_2' \to cC, C \to cC, S_2 \to Q_1\},$$
$$P_3 = \{S_3 \to S_3', S_3' \to S_3', S_3' \to Q_2, C \to C', C' \to C'', C'' \to D, D \to dD\}. \qquad \square$$

3 Summary

The present paper gives parallel communicating grammar systems (PC grammar systems) that generate the languages $K_1 = \{ a^n b^n c^n \mid n \geq 1 \}$, $K_2 = \{ a^n b^m c^n d^m \mid m, n \geq 1 \}$ and $K_3 = \left\{ ww \;\middle|\; w \in \{ a, b \}^+ \right\}$, but need less or/and less powerful components than systems published so far. Especially for the language K_2, the systems could be improved considerably.

References

[1] A. Chiţu. PC Grammar Systems Versus Some Non-Context-Free Constructions from Natural and Artificial Languages. In New Trends in Formal Languages, LNCS 1218, Springer, 1997, pp. 278–287.

[2] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, G. Păun. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach Sci. Publ., 1994.

[3] J. Dassow, G. Păun. *Regulated Rewriting in Formal Language Theory*, Springer, 1989.

[4] J. Dassow, G. Păun, G. Rozenberg. Grammar systems. In Handbook of Formal Languages, Springer, 1997, pp. 155–213.

[5] J. Dassow, B. Truthe, Gy. Vaszil. Einige parallel kommunizierende Grammatiksysteme zum Erzeugen spezieller nicht-kontextfreier Sprachen, Preprint, 2006.

[6] S. Dumitrescu, G. Păun. On the Power of Parallel Communicating Grammar Systems with Right-Linear Components. RAIRO Informatique Théorique et Applications/Theoretical Informatics and Applications 31 (1997) 4, pp. 331–354.

[7] M. A. Grando, V. Mitrana. A Possible Connection Between Two Theories: Grammar Systems and Concurrent Programming. In Proceedings of Grammar Systems Week 2004, MTA SZTAKI, Budapest, 2004, pp. 200–211.