Acta Cybernetica 12 (1996) 397-409.

Parallel Communicating Grammar Systems with Separated Alphabets*

Valeria MIHALACHE †

Abstract

The generative capacity of parallel communicating grammar systems is considered in the context that the component grammars have distinct terminal and nonterminal sets. In the regular case, this results in strictly more powerful systems in comparison to the classical ones. In the context-free case, characterization of recursively enumerable languages is obtained when λ-rules are allowed in non-centralized returning systems, deriving in the synchronized mode. Unsynchronized context-free systems with separated alphabets have the same power as the corresponding usual systems.

1 Introduction

One of the main trends of our days in several fields of computer science is to solve a complex problem by dividing it into subproblems, and then having it solved in a cooperative mode by several "processors". The concretization of this trend in grammar theory are the so-called *grammar systems.*

There are two basic models of grammar systems: *cooperating distributed* (CD, for short) grammar systems, which have been introduced in [2] (a former variant can be found in [7]; a particular case appears also in [1]), and *parallel communicating* (PC, for short) grammar systems, which have been introduced in [10].

Roughly speaking, a grammar system consists of several (Chomsky) grammars (called *components*) working together, towards generating a common language. In a CD grammar system the component grammars work in turn, on the same sentential form, only one being active at a given moment, according to a predefined protocol. In a PC grammar system the components work simultaneously, in a synchronized manner, each having its own sentential form and cooperating with the others by communication, which is done by request. The Artificial Intelligence counterpart

*Research supported by the Academy of Finland, project 11281

†Faculty of Mathematics, University of Bucharest, Str. Academiei 17, 70109 Bucharest, Romania. Current address: Turku Centre for Computer Science (TUCS), Lemminkäisenkatu 14 A, 4th Floor, 20520, Turku, Finland

of a CD grammar system is the ***blackboard model*** in problem solving, whereas to PC grammar system the *classroom model* corresponds (see [3]).

In the original definition, for parallel communicating grammar systems it is assumed that all the grammars have the same terminal and nonterminal sets. This is very convenient in terms of the classroom model. Thus, it is rather natural to assume that all the pupils in a classroom have similar background and similar abilities (that is, the associated grammars in the system share the same nonterminal set), and also that they are asked to perform similar tasks (in the corresponding formal modelisation, this implies the same terminal set for all the grammars). However, if one considers that not pupils are working towards solving a problem, but agents, instead, the working protocol being the same as in the classroom model, such assumptions are not natural anymore. Agents can have to perform totally different tasks, and they can have different skills.

Such a set-up can be modeled in the grammar systems framework by a slightly modification of the original variant of PC grammar systems. One can consider that any of the component grammars of the system has its own terminal and nonterminal sets ([3], [9]). The main difference between these systems and the usual ones is that here the same letter can act as terminal symbol in one grammar and nonterminal in another one. In the regular case, PC grammar systems modified like that are proved to be more powerful than systems of the initial form. Furthermore, in the context-free case, characterization of recursively enumerable languages is obtained.

2 Preliminary definitions

Throughout this paper, we use the notation and basic results of formal language theory from [4], [11] ; for grammar systems notions we refer to [3], [5]. We specify here only some notation.

For an alphabet V, V^* denotes the free monoid generated by V; the empty string is denoted by λ, $|x|$ is the length of $x \in V^*$ and $|x|_U$ is the number of occurrences in $x \in V^*$ of symbols of $U \subseteq V$. The classes of regular, context-free, type-0 grammars and matrix grammars with appearance checking are denoted by REG, CF, RE, MAT_{ac}, respectively. Unless otherwise specified, we consider in this paper only generative tools without λ-productions.

For a class X of generative mechanisms, the family of languages generated by elements of X is denoted by $\mathbf{L}(X)$.

Definition 1 *Let $n \geq 1$ be a natural number. A* parallel communicating grammar system of degree n with separated alphabets *(PC grammar system* of type s, *for short) is an $(n+1)$-tuple*

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

where $K = \{Q_1, Q_2, \ldots, Q_n\}$ and

$$G_i = (N_i \cup K, T_i, P_i, S_i), 1 \leq i \leq n,$$

are usual Chomsky grammars (the sets N_i, T_i, K being mutually disjoint, for any $i, 1 \leq i \leq n$).

We write $V_i = N_i \cup T_i \cup K$ and $V_\Gamma = \bigcup_{i=1}^{n}(N_i \cup T_i) \cup K$. The grammars $G_i, 1 \leq i \leq n$, are called *components* of the system, and the elements of K are called *query symbols*; their indices, $1, \ldots, n$, point to the components $G_1, \ldots, G_n$, respectively.

Remark that the definiton of PC grammar systems of type s does not require $N_i \cap T_j = \emptyset$ for $1 \leq i, j \leq n, i \neq j$.

The convention throughout this paper is to denote the start symbol and the production set of a component of a system with the same indices as the grammar component is denoted. This convention holds for query symbols, too, so we do not need to specify in details the set of query symbols for a given system.

The derivation in PC grammar systems of type s is defined in a similar manner as for usual PC grammar systems, that is

Definition 2 *Given a PC grammar system $\Gamma = (K, G_1, \ldots, G_n)$ as above, for two n-tuples $(x_1, x_2, \ldots, x_n), (y_1, y_2, \ldots, y_n), x_i, y_i \in V_i^*, 1 \leq i \leq n$, $x_1 \notin T_1^*$, we write $(x_1, \ldots, x_n) \Longrightarrow (y_1, \ldots, y_n)$ if one of the next two cases holds:*

(i) $|x_i|_K = 0, 1 \leq i \leq n$, and for each i, $1 \leq i \leq n$, we have $x_i \Longrightarrow y_i$ in the grammar G_i, or $x_i \in T_i^*$ and $x_i = y_i$;

(ii) there is an i, $1 \leq i \leq n$, such that $|x_i|_K > 0$; then for each such i, we write $x_i = z_1 Q_{i_1} z_2 Q_{i_2} \ldots z_t Q_{i_t} z_{t+1}, t \geq 1$, for $z_j \in V_\Gamma^*, |z_j|_K = 0, 1 \leq j \leq t+1$; if $|x_{i_j}|_K = 0, 1 \leq j \leq t$, then $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$, providing that $y_i \in V_i^*$, [and $y_{i_j} = S_{i_j}, 1 \leq j \leq t$]; when, for some j, $1 \leq j \leq t, |x_{i_j}|_K \neq 0$, then $y_i = x_i$; for all i, $1 \leq i \leq n$, for which y_i is not specified above, we have $y_i = x_i$.

Point (i) defines (componentwise) *derivation steps*, whereas point (ii) defines *communication steps.* In a communication operation, when the communicated string x_j replaces the query symbol Q_j, we say that Q_j is satisfied. The communication has priority over the effective rewriting. If some query symbols are not satisfied at a given communication step, then they will have to be satisfied at a next one. No rewriting is possible when at least one query symbol is present.

The work of a PC grammar system with separated alphabets is blocked in three cases: (1) when a component x_i of the current n-tuple $(x_1, \ldots, x_n)$ (sometimes we shall call it a *configuration*) is not terminal with respect to G_i, but no rule of G_i can be applied to x_i, or (2) when a *circular query* appears, that is G_{i_1} introduces Q_{i_2}, G_{i_2} introduces Q_{i_3}, and so on, until $G_{i_{k-1}}$ introduces Q_{i_k} and G_{i_k} introduces Q_{i_1} (because only strings without query symbols can be communicated), or (3) when after satisfying a query, the sentential form of a grammar is not a string over the alphabet of that grammar, that is, in case (*ii*) we have $z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1} \notin V_i^*$ (we recall that for usual PC grammar systems case (3) does not appear).

Definition 3 *The language generated by a PC grammar system Γ as above is*

$$L(\Gamma) = \{x \in T_1^* \mid (S_1, S_2, \ldots, S_n) \stackrel{*}{\Longrightarrow} (x, \alpha_2, \ldots, \alpha_n), \alpha_i \in V_\Gamma^*, 2 \leq i \leq n\}.$$

Observe that due to the definition of the language generated by a PC grammar system of type s we have that the terminal set of the system is actually the same as the terminal set of the first component of the system. From the definition of the derivation relation it follows that once the sentential form of G_1 has become a terminal string, the derivation cannot continue anymore in the system.

Just as in the case of usual PC grammar systems, one distinguishes several variants.

Definition 4 *If in Definition 2 only grammar G_1 is allowed to introduce query symbols, then we say that Γ is a* centralized *PC grammar system of type s; in contrast, the unrestricted case is called* non-centralized.

A PC grammar system of type s is said to be returning *(to axiom) if, after communicating, each component returns to axiom. A PC grammar system of type s is* non-returning *if in point (ii) of Definition 2, the brackets,*

$$[and\ y_{i_j} = S_{i_j}, 1 \le j \le t],$$

are omitted.

A PC grammar system is said to be regular, context-free, λ-free, etc., when the rules of its components are of these types.

For $n \ge 1$ and $X \in \{REG, CF\}$, we shall denote the families of languages generated by *non-returning centralized, non-returning non-centralized, returning centralized*, and *returning non-centralized*, respectively, PC grammar systems of type s, of degree at most n and with components of type X by

$$\mathbf{L}(NCPC_nX, s)\ ;\ \ \mathbf{L}(NPC_nX, s)\ ;\ \ \mathbf{L}(CPC_nX, s)\ ;\ \ \mathbf{L}(PC_nX, s).$$

When an arbitrary number of components is considered, we shall use $*$ instead of n.

If we still require in point (ii) of Definition 2 that those x_{i_j} which are to be communicated must be terminal strings in the grammars whose sentential forms they are, that is $x_{i_j} \in T_{i_j}^*$, then we say that Γ derives in the *terminal* mode. The language generated by Γ in this way is denoted by $L_T(\Gamma)$ and the family corresponding to $\mathbf{L}(Y_nX, s), \mathbf{L}(Y_*X, s)$ as above is denoted by $\mathbf{L}_T(Y_nX, s), \mathbf{L}_T(Y_*X, s)$. Here and in the sequel Y ranges over $\{NCPC, NPC, CPC, PC\}$ or some specified subset of it.

If we replace point (i) in Definition 2 by

$(i')|x_i|_K = 0, 1 \le i \le n$ *and, for each $i, 1 \le i \le n$, we have either $x_i \Rightarrow y_i$ in grammar G_i, or $x_i = y_i$*, then, just as in the case of usual PC grammar systems, we get a *PC grammar system of type s deriving in an unsynchronized manner.*

Denote by $L_U(\Gamma)$ the language generated by Γ in this way. The family of languages generated by unsynchronized PC grammar systems of type s corresponding to a family $\mathbf{L}(Y_nX, s), \mathbf{L}(Y_*X, s)$ as above is denoted by $\mathbf{L}_U(Y_nX, s), \mathbf{L}_U(Y_*X, s)$.

In order to illustrate the difference between usual PC grammar systems and the ones studied here, let us consider an example.

Example: Let $\Gamma = (K, G_1, G_2)$, be the returning non-centralized PC grammar system with

$$\begin{aligned} G_1 &= (\{S_1, A, B, Y\}, \{X, a, b, c\}, \\ &\quad \{S_1 \to XA, A \to XA, S_1 \to aB, B \to aB, B \to aQ_2, Y \to b\}, S_1) \\ G_2 &= (\{S_2, X, A\}, \{Y, c\}, \{S_2 \to YS_2, S_2 \to YQ_1, X \to c, A \to c\}, S_2) \end{aligned}$$

If we start a derivation by using in G_1 the production $S_1 \to aB$, then we have
either $(S_1, S_2) \overset{*}{\Rightarrow} (a^{k+1}Q_2, Y^{k+1}Z)$, $k \geq 1, Z \in \{Q_1, S_2\}$,
and then the derivation is blocked due to circular query, if Z is Q_1, or due to $S_2 \notin N_1 \cup T_1$, when attempting to satisfy the query in G_1, if Z is S_2,
or $(S_1, S_2) \overset{*}{\Rightarrow} (a^{k+1}B, Y^{k+1}Q_1)$, $k \geq 0$,
and then the derivation is blocked because $B \notin N_2 \cup T_2$, and hence we cannot satisfy the query in G_2.

The only successful derivation is the one which starts as
$(S_1, S_2) \overset{*}{\Rightarrow} (X^{k+1}A, Y^{k+1}Q_1) \Rightarrow (S_1, Y^{k+1}X^{k+1}A) \overset{*}{\Rightarrow} (a^lQ_2, Y^{k+1}\alpha) \Rightarrow (a^lY^{k+1}\alpha, S_2)$,
where $k \geq 1$ and α is obtained from the string $X^{k+1}A$ by replacing some of the X-s and/or A with c, and $l = |\alpha|_c$, if $\alpha \notin T_2^*$, or $l \geq k+2$, if $\alpha \in T_2^*$ (i.e. $\alpha = c^{k+2}$). If $\alpha \notin T_2$, in order to rewrite it as a terminal string, after a number of steps G_2 must ask for the sentential form of G_1. But this sentential form contains symbols a, which are not in the alphabet of G_2, and hence the query would not be satisfied. This implies that it must be the case $\alpha \in T_2$, and then the configuration above is

$$(a^lY^{k+1}c^{k+2}, S_2), l \geq k+2.$$

Because, as we have made the observation, G_2 cannot accept a sentential form of G_1 containing symbol a, then the derivation has to end as

$$(a^lY^{k+1}c^{k+2}, S_2) \overset{*}{\Rightarrow} (a^lb^{k+1}c^{k+2}, Y^{k+1}Z).$$

Thus we have that

$$L(\Gamma) = \{a^{k+1+s}b^kc^{k+1} \mid s \geq 0, k \geq 1\},$$

a language which is not context-free. Note that what increases the power of the system (for the usual PC grammar systems we have $\mathbf{L}(PC_2REG) \subseteq \mathbf{L}(CF)$, see [13]) was the possibility of a component to rewrite symbols which are considered terminals in another one, as well as the restriction that a communicated string has to be a string over the alphabet of the grammar which required it.

3 Generative Capacity

We first present some general properties for parallel communicating grammar systems of type s, which are true also in case of usual parallel communicating grammar systems.

Lemma 1 *For any $Y \in \{PC, CPC, NPC, NCPC\}$ and for any class X of grammars, we have*

(i) $\mathbf{L}(Y_1X, s) = \mathbf{L}(X)$;
$\mathbf{L}_U(Y_1X, s) = \mathbf{L}(X)$;
$\mathbf{L}_T(Y_1X, s) = \mathbf{L}(X)$;

(ii) $\mathbf{L}(Y_nX, s) \subseteq \mathbf{L}(Y_{n+1}X, s), n \geq 1$,
$\mathbf{L}_U(Y_nX, s) \subseteq \mathbf{L}_U(Y_{n+1}X, s), n \geq 1$,
$\mathbf{L}_T(Y_nX, s) \subseteq \mathbf{L}_T(Y_{n+1}X, s), n \geq 1$,

(iii) $\mathbf{L}(CPC_nX, s) \subseteq \mathbf{L}(PC_nX, s)$; $\mathbf{L}(NCPC_nX, s) \subseteq \mathbf{L}(NPC_nX, s), n \geq 1$;
$\mathbf{L}_U(CPC_nX, s) \subseteq \mathbf{L}_U(PC_nX, s)$; $\mathbf{L}_U(NCPC_nX, s) \subseteq \mathbf{L}_U(NPC_nX, s)$, $n \geq 1$;
$\mathbf{L}_T(CPC_nX, s) \subseteq \mathbf{L}_T(PC_nX, s)$; $\mathbf{L}_T(NCPC_nX, s) \subseteq \mathbf{L}_T(NPC_nX, s)$, $n \geq 1$;

(iv) $\mathbf{L}(CPC_*X, s) \subseteq \mathbf{L}(PC_*X, s)$; $\mathbf{L}(NCPC_*X, s) \subseteq \mathbf{L}(NPC_*X, s)$;
$\mathbf{L}_U(CPC_*X, s) \subseteq \mathbf{L}_U(PC_*X, s)$; $\mathbf{L}_U(NCPC_*X, s) \subseteq \mathbf{L}_U(NPC_*X, s)$;
$\mathbf{L}_T(CPC_*X, s) \subseteq \mathbf{L}_T(PC_*X, s)$; $\mathbf{L}_T(NCPC_*X, s) \subseteq \mathbf{L}_T(NPC_*X, s)$.

Proof: Directly from definitions. □

Each usual PC grammar system can be considered a PC grammar system of type s (we simply skip N, T when writing $\Gamma = (N, K, T, G_1, \dots, G_n)$), hence we also have

Lemma 2 *For any $Y \in \{PC, CPC, NPC, NCPC\}$ and for any class X of grammars,*

(i) $\mathbf{L}(Y_nX) \subseteq \mathbf{L}(Y_nX, s), n \geq 1$;

(ii) $\mathbf{L}_U(Y_nX) \subseteq \mathbf{L}_U(Y_nX, s), n \geq 1$;

(iii) $\mathbf{L}_T(Y_nX) \subseteq \mathbf{L}_T(Y_nX, s), n \geq 1$;

Just as in the case of usual PC grammar systems, we have relations between the families of generated languages, when considering various modes of derivation.

Lemma 3 *(i)* $\mathbf{L}_U(Y_nX, s) \subseteq \mathbf{L}(Y_nX, s)$, *for any class X of grammars allowing chain rules (that is rules of the form $A \to B$) and for any $Y \in \{CPC, PC, NCPC, NPC\}$*;

(ii) $\mathbf{L}_T(Y_nX, s) \subseteq \mathbf{L}(Y_nX, s)$, *for any class X of grammars and for any $Y \in \{CPC, NCPC\}$.*

Proof: The proofs are entirely the same as for usual PC grammar systems, [3]. □

We next survey the properties known so far about regular PC grammar systems with separated alphabets. For the proofs we refer to [9].

Proposition 1 *(i) The family* $\mathbf{L}(PC_3REG, s)$ *contains one-letter non-regular languages.*

(ii) The families $\mathbf{L}(NCPC_2REG, s), \mathbf{L}(NPC_2REG, s)$ *contain one-letter non-regular languages.*

(iii) The families $\mathbf{L}_U(NCPC_2REG, s), \mathbf{L}_U(NPC_2REG, s)$ *contain non-semi-linear languages.*

(iv) The family $\mathbf{L}_T(CPC_2REG, s)$ *contains non-semi-linear languages.*

Corollary 1 $\mathbf{L}(CPC_nREG) \subset \mathbf{L}(CPC_nREG, s)$, *strict inclusion, for any* $n \geq 2$.

So just as in the case of cooperating distributed grammar systems, by considering distinct terminal sets for the grammar components of the system, also in the case of PC grammar systems, the generative power is increased (at least in the regular centralized returning case).

But even if centralized returning PC grammar systems with regular components of type s are able to generate non-finite index matrix languages, we still can find an upper-bound for the languages generated by them among the regulated rewriting tools with context-free rules. More exactly, we have

Theorem 1 *(i)* $\mathbf{L}(CPC_*REG, s) \subseteq \mathbf{L}(MAT_{ac})$.

(ii) $\mathbf{L}_U(CPC_*REG, s) \subseteq \mathbf{L}(MAT_{ac})$.

(iii) $\mathbf{L}_T(CPC_*REG, s) \subseteq \mathbf{L}(MAT_{ac})$.

One can observe that although the preceding theorem is for the regular PC grammar systems, it is true as well for the right linear case, and the proof is entirely the same.

As we shall prove in the following, for unsynchronized derivation we can actually find a more specific relation. First, we have the theorem

Theorem 2 $\mathbf{L}_U(CPC_2REG, s) - \mathbf{L}(REG) \neq \emptyset$.

Proof: Consider the following PC grammar system of type s with regular components

$$\Gamma = (K, G_1, G_2),$$

where

$$\begin{aligned} G_1 &= (\{S_1, A, B\}, \{a, b\}, \{S_1 \to aQ_2, A \to aQ_2, B \to b, A \to a\}, S_1), \\ G_2 &= (\{S_2, B\}, \{A\}, \{S_2 \to AB\}, S_2), \end{aligned}$$

and consider the derivation mode to be the unsynchronized one.

Then a terminal derivation has to proceed as follows:

$$\begin{aligned}(S_1,S_2) &\overset{*}{\Rightarrow} (aQ_2,AB)\Rightarrow(aAB,S_2)\overset{*}{\Rightarrow}(a^2Q_2X,AB)\Rightarrow(a^2ABX,S_2)\\ &\overset{*}{\Rightarrow} \dots \overset{*}{\Rightarrow}(a^kQ_2\alpha,AB)\Rightarrow(a^kAB\alpha,S_2)\overset{*}{\Rightarrow}(a^{k+1}b^k,\beta)\end{aligned}$$

where $X\in\{b,B\}$, $\alpha\in\{b,B\}^*,|\alpha|=k-1$ and $\beta\in\{S_2,AB\}$.

We then have

$$L(\Gamma)=\{a^{k+1}b^k \mid k\geq 1\},$$

which is not a regular language. □

As a corollary, we obtain that also in the unsynchronized derivation, centralized returning PC grammar systems are more powerful when considering distinct sets of terminal and non-terminal symbols then in the case when we do not.

Corollary 2 $\mathbf{L}_U(CPC_nREG)\subset\mathbf{L}_U(CPC_nREG,s)$, *strict inclusion, for any* $n\geq 2$.

Proof: The inclusion is by Lemma 2, and the strictness of it follows from the above theorem and from $\mathbf{L}_U(CPC_*REG)=\mathbf{L}(REG)$, which is known from [3]. □

Our intention in the following is to present other properties concerning PC grammar systems of type s deriving in the unsynchronized mode. We need to recall the following definition.

Definition 5 *Let* $\Gamma=(N,K,T,G_1,G_2,\dots,G_n)$ *be a usual parallel communicating grammar system. We say that* Γ *is* with multiple queries *if there is a component of* Γ *with a production* $A\rightarrow\alpha Q_i\beta Q_i\gamma,\alpha,\beta,\gamma\in(N\cup K\cup T)^*,i\in\{1,\dots,n\}$. *Otherwise, we say that* Γ *is* without multiple queries.

The class of such grammar systems is denoted by WY_nX, for $Y\in\{PC, CPC, NPC, NCPC\}$, $n\geq 1$, X a class of grammars.

Theorem 3 *For any* $Y\in\{CPC,PC,NCPC,NPC\}$ *and for any* $n\geq 1$,

(i) $\mathbf{L}_U(Y_nREG,s)\subseteq\mathbf{L}_U(WY_nCF)$;

(ii) $\mathbf{L}_U(Y_nCF,s)=\mathbf{L}_U(Y_nCF)$.

(iii) $\mathbf{L}_U(WY_nCF,s)=\mathbf{L}_U(WY_nCF)$.

Proof: To prove point (i), take a PC grammar system of type s with regular components

$$\Gamma=(K,G_1,G_2,\dots,G_n),$$

with $G_i=(N_i,T_i,P_i,S_i)$, for any $i, 1\leq i\leq n$.

Denote

$$V=\bigcup_{i=1}^{n}(N_i\cup T_i).$$

For each symbol $\alpha \in V$, consider a new symbol α', denote by V' their set and define the substitution h as

$$\begin{aligned} h(\alpha) &= \{\alpha, \alpha'\}, \text{ for } \alpha \in T_1, \\ h(\alpha) &= \{\alpha'\}, \text{ for } \alpha \in V - T_1, \\ h(Q_i) &= \{Q_i\}, \text{ for } 1 \le i \le n. \end{aligned}$$

Construct the PC grammar system

$$\Gamma' = (V', K, T_1, (P_1', S_1'), \ldots, (P_n', S_n')),$$

where

$$P_i' = \{A' \to y \mid A \to x \in P_i, y \in h(x)\}, \text{ for any } 1 \le i \le n.$$

Note that even if we have started from a PC grammar system Γ with regular components, because this is of type s, the resulted system, Γ', is with context-free components and not with regular. This happens because we can have in a component $G_i, i \ge 1$ a production $A \to BC$, where B is a terminal symbol with respect to G_i but is not a terminal symbol with respect a grammar to which it will communicate a string containing that B.

Moreover, note that if Γ is returning, centralized, non-returning or non-centralized, then Γ' is of the same type.

Because in any production of any grammar at most one query symbol can appear, we have that Γ' is without multiple queries.

One can see that $L_U(\Gamma) = L_U(\Gamma')$, and thus point (i) follows.

For point (ii), we have $\mathbf{L}_U(Y_nCF) \subseteq \mathbf{L}_U(Y_nCF, s)$ by Lemma 2. To prove the reverse inclusion, we only need to observe that the same construction that we have considered for the proof of point (i) transforms a context-free PC grammar system of type s into a corresponding usual context-free PC grammar system.

Point (iii) is a consequence of the relation in point (ii), by the observation that the construction we have considered does not introduce multiple queries. □

Corollary 3 $\mathbf{L}_U(WCPC_*CF, s) = \mathbf{L}(CF)$.

Proof: It is simply a consequence of the preceding theorem, point (iii), by Theorem 1 of [8], which states that $\mathbf{L}_U(WCPC_*CF) = \mathbf{L}(CF)$. □

Now we can improve the relation obtained in Theorem 1, for the case of unsynchronized derivation. That is, we have

Theorem 4 $\mathbf{L}(REG) \subset \mathbf{L}_U(CPC_*REG, s) \subseteq \mathbf{L}(CF)$.

Proof: The first inclusion is from Lemma 1 and from Theorem 2. The second inclusion is a consequence of the preceding theorem, point (i), by Theorem 1 of [8]. □

Note that once again parallel communicating grammar systems of type s are more powerful, but still not "too powerful" (from the generative capacity point of view) than usual PC grammar systems, in case of regular components, because we have $\mathbf{L}_U(CPC_*REG) = \mathbf{L}(REG)$. But in case of context-free components, for unsynchronized derivation, systems of type s are only as powerful as usual systems are.

It is known, [8], that matrix languages can be generated by usual returning non-centralized PC grammar systems with context-free components. When separated terminal and nonterminal alphabets are considered for the components of the system, one can simulate matrix grammars with appearance checking.

Theorem 5 $\mathbf{L}(MAT_{ac}) \subseteq \mathbf{L}(PC_*CF, s)$.

Proof: The proof bears resemblance with the corresponding one in [8]. Let

$$G = (N, T, S, M, F)$$

be a context-free matrix grammar with the appearance checking set F. It is known ([4]) that for each matrix grammar there is an equivalent matrix grammar, of the same type, in the 2-normal form, that is with

$$N = \{S\} \cup N_1 \cup N_2, \quad N_1 \cap N_2 = \emptyset, \quad S \notin N_1 \cup N_2,$$

and each matrix of M has one of the following forms:

$$\begin{array}{ll} (i) & (S \to AX), \ A \in N_1, \ X \in N_2 \\ (ii) & (A \to \alpha, X \to Y), \ A \in N_1, \ \alpha \in (N_1 \cup T)^+, X, Y \in N_2, \\ (iii) & (A \to \alpha, X \to a), \ A \in N_1, \ \alpha \in (N_1 \cup T)^+, X \in N_2, a \in T, \\ (iv) & (S \to x), x \in T^*. \end{array}$$

Moreover, the productions of F are only of the form $A \to \alpha$, with $A \in N_1, \alpha \in (N \cup T)^*$.

Let $P_1(M)$ be the set of matrices of type *(i)*, let $P_2(M)$ be the set of matrices of types *(ii)*, *(iii)* and let r be the cardinality of $P_2(M)$. A matrix of $P_2(M)$ will be denoted in the following by

$$m_k : \ (A_k \to \alpha_k, B_k \to C_k) \ , \ 1 \le k \le r.$$

Denote

$$N' = N \cup \{S', W, V, Z, L_1, L_2, L_3\} \cup \{S_s, S_{a1}, S_{a2}\} \cup \{S_{1k}, S_{2k} \mid k = 1, \dots, n\}$$

$(S', W, V, Z, L_1, L_2, L_3$ are new symbols). We construct the PC grammar system

$$\Gamma = (K, G_s, G_{11}, G_{21}, G_{12}, G_{22}, \dots, G_{1r}, G_{2r}, G_{a1}, G_{a2})$$

as follows:

$$
\begin{aligned}
P_s &= \{S_s \rightarrow x \mid (S_s \rightarrow x) \in M, x \in T^*\} \cup \\
&\quad \cup \{S_s \rightarrow S', S' \rightarrow Q_{2k} \mid k = 1, \ldots, r\} \cup \\
&\quad \cup \{S_s \rightarrow AB \mid (S \rightarrow AB) \in P_1(M)\} \cup \\
&\quad \cup \{X \rightarrow X \mid X \in N_2\}, \\
P_{1k} &= \{S_{1k} \rightarrow W, W \rightarrow Q_s\} \cup \\
&\quad \cup \{A_k \rightarrow \alpha_k \mid m_k : (A_k \rightarrow \alpha_k, B_k \rightarrow C_k)\} \cup \\
&\quad \cup \{X \rightarrow Z \mid X \in N_1 \cup N_2\} \cup \\
&\quad \cup \{V \rightarrow Q_{1k}\}, \text{ for each } k = 1, 2, \ldots, r, \\
P_{2k} &= \{S_{2k} \rightarrow V, V \rightarrow Q_{1k}\} \cup \\
&\quad \cup \{B_k \rightarrow C_k \mid m_k : (A_k \rightarrow \alpha_k, B_k \rightarrow C_k)\} \cup \\
&\quad \cup \{X \rightarrow Z \mid X \in N_1 \cup N_2\}, \text{ for each } k = 1, 2, \ldots, r, \\
P_{a1} &= \{S_{a1} \rightarrow Q_{21}Q_{22}\ldots Q_{2r}\} \cup \{V \rightarrow V\}, \\
P_{a2} &= \{S_{a2} \rightarrow L_1, L_1 \rightarrow L_2, L_2 \rightarrow L_3, L_3 \rightarrow L_1Q_{21}Q_{22}\ldots Q_{2r}\}.
\end{aligned}
$$

The terminal sets of the components grammars of Γ are defined as

$$T_s = T_{a1} = T_{a2} = T_{2k} = T, \text{ for any } k = 1, 2, \ldots, r,$$

while for any $k = 1, 2, \ldots, r$,

$$
T_{1k} = \begin{cases} T \cup N_2 \cup N_1 - \{A_k \mid m_k : (A_k \rightarrow \alpha_k, B_k \rightarrow C_k)\}, & \text{if this occurrence of the production } A_k \rightarrow \alpha_k \in F \\ T, & \text{otherwise.} \end{cases}
$$

As for the nonterminal sets of the components of Γ, they are defined as

$$
\begin{aligned}
N_s &= N' - T_s, \\
N_{1k} &= N' - T_{1k}, \text{ for any } k = 1, \ldots, r, \\
N_{2k} &= N' - T_{2k}, \text{ for any } k = 1, \ldots, r, \\
N_{a1} &= N' - T_{a1}, \\
N_{a2} &= N' - T_{a2}.
\end{aligned}
$$

One can verify that $L(\Gamma) = L(G)$, and therefore the theorem follows. □

As an immediate corollary of the above theorem, characterization of recursively enumerable languages results when λ-productions are allowed in the system.

Corollary 4 $\mathbf{L}(PC_*CF^\lambda, s) = \mathbf{L}(RE)$, *where the notation* PC_*CF^λ *stands for returning non-centralized PC grammar systems with* λ*-rules.*

The similarity between PC grammar systems with separated alphabets and usual PC grammar systems let us think that any proof of a relation between two classes of usual PC grammar systems can be adapted as to result in a relation between the corresponding classes of PC systems of type s. More precisely, we conjecture that if $\mathbf{L}(Y_nCF) \subseteq \mathbf{L}(Y'_mCF)$ for two classes Y, Y' of PC grammar systems, $m, n, \geq 1$, then $\mathbf{L}(Y_nCF, s) \subseteq \mathbf{L}(Y'_mCF, s)$.

In particular, by [6], [12],

$$\mathbf{L}(NPC_*CF, s) \subseteq \mathbf{L}(PC_*CF, s)$$

would result.

References

[1] A. Atanasiu, V. Mitrana, The Modular Grammars, *Internat. J. Comp. Math.* 30 (1989), 101-122

[2] E. Csuhaj-Varjú, J. Dassow, On Cooperating Distributed Grammar Systems, *J. Inf. Processing and Cybern. EIK*, 26 (1990), 49-63

[3] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach Science Publishers Ltd, London, 1994

[4] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, Heidelberg, 1989

[5] J. Dassow, Gh. Păun, G. Rozenberg, Grammar Systems, in G. Rozenberg, A. Salomaa (eds.), *The Handbook of Formal Languages*, Springer-Verlag, to appear

[6] S. Dumitrescu, Non-returning parallel communicating grammar systems can be simulated by returning systems, *Theoretical Computer Science*, 165 (1996), 463-474.

[7] R. Meersman, G. Rozenberg, Cooperating Grammar Systems, in *Proc. MFCS'78 Symp. LNCS 64*, Springer-Verlag, Berlin, 1978, 364-374

[8] V. Mihalache, Matrix Grammars versus Parallel Communicating Grammar Systems, in Gh. Păun (ed.), *Mathematical Aspects of Natural and Formal Languages*, World Scientific, 1994, 293-318

[9] V. Mihalache, Terminal versus Non-terminal Symbols in Parallel Communicating Grammar Systems, *Revue Roumaine de Mathématiques Pures et Appliquées*, to appear

[10] Gh. Păun, L. Sântean, Parallel Communicating Grammar Systems: the Regular Case, *Ann. Univ. Buc., Ser. Matem.-Inform.*, 38 (1989), 55-63

[11] A. Salomaa, *Formal Languages*, Academic Press, New York, London, 1973

[12] Gy. Vaszil, On Simulating Non-Returning PC Grammar Systems with Returning Systems, *submitted for publication*

[13] S. Vicolov, Non-centralized Parallel Grammar Systems, *Stud. Cerc. Mat.*, 44 (1992), 455-462