On the Generative Capacity of Parallel Communicating Grammar Systems with Regular Components

Valeria Mihalache

Turku Centre for Computer Science

TUCS Technical Reports

No 8, May 1996

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

May 1996

ISBN 951-650-749-2
ISSN 1239-1891

Abstract

The paper considers the simplest class of parallel communicating grammar systems, namely with regular components, which yet seems to be not enough investigated. It is proved that in the centralized returning case (deriving in the usual mode) and in the non-centralized non-returning case with terminal derivation the systems are not "too powerful", more exactly they can be simulated by finite index matrix grammars.

1 Introduction

Grammar systems have been proved to be a powerful tool, appealing from both formal languages and Artificial Intelligence point of view.

There are two basic classes of grammar systems, namely *cooperating distributed grammar systems*, which have been introduced in [2], (a former variant of them can be found in [6]; a particular case appears also in [1]), and *parallel communicating grammar systems*, which have been introduced in [8]. Even though they are not very "old", grammar systems have captured the interest of many people, as can be seen from the impressive number of papers concerning this subject. We only want to mention the monograph [3], where most of the results on grammar systems known at the middle of 1992 can be found.

The present paper addresses parallel communicating grammar systems with regular components, deriving in the usual or in the terminal mode. Some classes of such grammar systems are compared with finite index matrix grammars, more specifically it is proved that the former classes are not more powerful than the latter grammars. It is not the first attempt to find relations between parallel communicating grammar systems and matrix grammars. Results on this topic can be also found in [7], but for systems with context-free components: the matrix grammars (without appearance checking) can be simulated by non-centralized returning parallel communicating grammar systems with context-free components.

The results outlined in the paper concern grammar systems without $\lambda-$ productions, but similar properties could be inferred in an obvious manner also for systems with $\lambda-$ productions.

From the results of this paper and from the results in [7], one can also obtain relations between some classes of parallel communicating grammar systems with regular and with context-free components.

2 Preliminary definitions

Throughout this paper, we use the notation and basic results of formal language theory from [4], [9] ; for grammar systems notions we refer to [3]. 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, linear, context-free, finite index matrix grammars and matrix grammars are denoted by $REG, LIN, CF, MAT_{fin}, MAT$, respectively. We consider in this paper only generative tools without $\lambda-$ 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 is an $n+3$-tuple*

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

where N is a nonterminal alphabet, T is a terminal alphabet, $K = \{Q_1, Q_2, \ldots$ $\ldots, Q_n\}$ (these sets are mutually disjoint), and

$$G_i = (N \cup K, T, P_i, S_i), 1 \leq i \leq n,$$

are usual Chomsky grammars.

We write $V_\Gamma = N \cup K \cup T$.

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.

The convention we consider 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 detail the set of query symbols for a given system.

Definition 2 *Given a PC grammar system $\Gamma = (N, K, T, G_1, \ldots, G_n)$ as above, for two n-tuples $(x_1, x_2, \ldots, x_n), (y_1, y_2, \ldots, y_n), x_i, y_i \in V_\Gamma^*, 1 \leq i \leq n$, 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 $x_i \in T^*$ 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 $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$ [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 is blocked in two cases: (1) when a derivation step must be performed and a component x_i of the current n-tuple $(x_1, \ldots, x_n)$ (sometimes we shall call it *configuration*) is not terminal with respect to G_i, but no rule of G_i can be applied to x_i, and (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, none of the query symbols can be satisfied).

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

$$\begin{aligned} L(\Gamma) \;=\; & \{x \in T^* \mid (S_1, S_2, \ldots, S_n) \stackrel{*}{\Longrightarrow} (x, \alpha_2, \ldots, \alpha_n), \alpha_i \in V_\Gamma^*, 2 \leq i \leq n, \\ & \text{and in this derivation there is no intermediate configuration} \\ & \text{having a terminal string as the sentential form of } G_1\}. \end{aligned}$$

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; in contrast, the unrestricted case is called* non-centralized.

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

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

are omitted.

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

For $n \geq 1$ and $X \in \{REG, LIN, CF\}$, we shall denote by

$$NCPC_nX,\ NPC_nX,\ CPC_nX,\ PC_nX,$$

the class of *non-returning centralized, non-returning non-centralized, returning centralized, and returning non-centralized*, respectively, grammar systems of degree at most n, with components of type X. The families of generated languages are denoted by $\mathbf{L}(Y_nX)$, respectively, where $Y \in \{NCPC, NPC,$

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

If we still require in point (ii) in Definition 2 that those x_{i_j} which are to be communicated must be terminal strings, that is $x_{i_j} \in T^*$, 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 $\mathbf{L}(Y_n X)$ as above is denoted by $\mathbf{L}_T(Y_n X)$.

3 Results

This section studies the generative capacity of parallel communicating grammar systems with regular components. The basic results provide some upper bounds for some of these classes. We make the observation that even if the results outlined concern the regular case, they are true for systems with right linear components as well.

Just for a global image on regular parallel communicating grammar systems, we start the section by recalling some properties already known. The numbers in parentheses which follow the properties are the numbers of theorems (corollaries) under which the properties are presented in [3].

Proposition 1

- $\mathbf{L}(CPC_2REG) \subset \mathbf{L}(CF)$, *strict inclusion* (Theorem 7.9)*;*
- $\mathbf{L}(CPC_nREG)$ *is incomparable with* $\mathbf{L}(CF)$, *for* $n \geq 3$ (Theorems 7.5, 7.7)*;*
- $\mathbf{L}(PC_2REG) \subseteq \mathbf{L}(CF)$ (Theorem 7.10)*;*
- *Each language in* $\mathbf{L}(CPC_*REG)$ *is semi-linear* (Theorem 7.11).

In the following we prove a result that improves the one outlined in [3], Theorem 7.11 (we know that any semi-linear language is letter equivalent with a finite index matrix language). That is, we have

Theorem 1 $\mathbf{L}(CPC_*REG) \subset \mathbf{L}(MAT_{fin})$, *strict inclusion.*

Proof: Take Γ a PC grammar system of degree $n, n \geq 1$. For $n = 1$, $L(\Gamma) \in \mathbf{L}(REG)$, hence the assertion is obvious. Thus, in the proof, we can consider only the case of PC grammar systems with at least two components.

Hence let

$$\Gamma = (N, K, T, G_1, G_2, \ldots, G_{n+1})$$

be a returning centralized regular PC grammar system, $n \geq 1$.

We denote $\mathcal{N} = \{[2], [3], \ldots, [n+1]\}$ and let $\mathcal{P}(\mathcal{N})$ be the set of nonempty subsets of $\mathcal{N}$. Let us consider $\mathcal{P}(\mathcal{N}) = \{M_1, M_2, \ldots, M_{2^n-1}\}$.

Also, let $\mathcal{P}_{i,q_i}$ be the set of permutations of the set M_i, represented as strings, for each $i, 1 \leq i \leq 2^n - 1, q_i = card(M_i)$. That is,

$$\mathcal{P}_{i,q_i} = \{y_{i,1}, y_{i,2}, \ldots, y_{i,q_i!}\},$$

$$y_{i,j} = [c_{m_1}][c_{m_2}] \ldots [c_{m_{q_i}}],$$

where $M_i = \{[c_1], [c_2], \ldots, [c_{q_i}]\}$.

(We write $y_{i,j}$ as above, instead of writing it as a function, in order to simplify the notation. The intended meaning for it is that a c_{m_l} is the value of the permutation for the argument $l, 1 \leq l \leq q_i$. Also for the sake of simplicity, we have not provided the items $[c_{m_l}]$ in $y_{i,j}$ with i, j, but they actually depend on them.)

To each permutation $y_{i,j} = [c_{m_1}][c_{m_2}] \ldots [c_{m_{q_i}}]$, we associate a string

$$x_{i,j} = [c_{m_1}]^{(i,j)}[c_{m_2}]^{(i,j)} \ldots [c_{m_{q_i}}]^{(i,j)}$$

and we consider

$$\mathcal{S}_{i,q_i} = \{x_{i,1}, x_{i,2}, \ldots, x_{i,q_i!}\},$$

for any $1 \leq i \leq 2^n - 1$.

We still consider for each $i, 1 \leq i \leq n+1$, $P'_i = P_i \cup \{X^{(i)} \to X^{(i)}\}$ and the morphism h_i defined on $N \cup T \cup \{X\}$ (where X is a new symbol), as

$$\begin{aligned} h_i(a) &= a, \, for\, any\, a \in T \\ h_i(A) &= A^{(i)}, \, for\, any\, A \in N \cup \{X\} \end{aligned}$$

For saving space, we denote

$$\begin{aligned} h_i(A \to aB) &= A^{(i)} \to aB^{(i)} \\ h_i(A \to a) &= A^{(i)} \to a \end{aligned}$$

Also consider the morphism g_i defined on $P'_i \cup \{A \to (0) \mid A \to \alpha \in P'_i\}$ as

$$\begin{aligned} g_i(A \to aB) &= \langle A \rangle^{(i)} \to \langle B \rangle^{(i)} \\ g_i(A \to a) &= \langle A \rangle^{(i)} \to X^{(i)} \\ g_i(X^{(i)} \to X^{(i)}) &= X^{(i)} \to X^{(i)} \\ g_i(A \to (0)) &= A^{(i)} \to (0), \, for\, any\, A \to \alpha \in P_i \\ g_i(X^{(i)} \to (0)) &= X^{(i)} \to (0). \end{aligned}$$

To prove that $L(\Gamma)$ is a matrix language, we will simulate the derivations in the grammar system Γ by a derivation in a matrix grammar whose sentential forms are of the type

$$h_1(\alpha_1)h_{l_1}(\alpha_{l_1})h_{l_2}(\alpha_{l_2})\dots h_{l_p}(\alpha_{l_p})\beta x, p \geq 0,$$

where either $x = [a_1]^{(i,s)}[a_2]^{(i,s)}\dots[a_p]^{(i,s)}(= x_{i,s}) \in \mathcal{S}_{i,p}$ for that i for which $M_i = \{[l_1],\dots,[l_p]\} \in \mathcal{P}(\mathcal{N})$, or $x = \langle l_p \rangle$ and then $p = 1$; $\beta \in \{h_i(X), h_i(A) | A \in N, 2 \leq i \leq n+1\}^*, |\beta| = n - p$ and $\{l_1, l_2, \dots, l_p\} \cap \{j \mid$ there exists a symbol superscribed with (j) in $\beta, 2 \leq j \leq n+1\} = \emptyset$.

Such a sentential form is meant to correspond to a derivation step in the grammar system at the moment such that α_1 is the sentential form of G_1, α_{l_j} is the sentential form of G_{l_j}, for any $j, 1 \leq j \leq p$, and during the following steps of the derivation the component G_{l_1} will be the first to be queried a number of times, at least once, afterwards G_{l_2} will be queried at least once, then maybe again G_{l_1}, or G_{l_2}, then G_{l_3} will be queried at least once, then again maybe G_{l_1} or G_{l_2}, or G_{l_3}, and so on, finally G_{l_p} will be queried at least once, then maybe again some of the grammars $G_{l_1}, \dots, G_{l_{p-1}}$, or G_{l_p}, and the derivation will be ended by modifying the sentential form of G_1 only by rewriting rules involving no communication.

The order in which the grammars are queried by G_1 is preserved by the string x, which imposes the order in which the strings α_{l_j} appear. The strings $h_1(\alpha_1)$, $h_{l_1}(\alpha_{l_1}), \dots, h_{l_p}(\alpha_{l_p})$ correspond to the sentential forms of the grammars $G_1, G_{l_1}, \dots, G_{l_p}$ of the system, respectively. They are superscribed by the morphisms h to make distinction between the nonterminals to be rewritten when one intends to simulate a rewriting step in a component of the system.

The string β is for the derivation in the components of the system which are not queried (or are not queried anymore, from the present step to the end of the derivation; and for we will use productions of the form $\langle A \rangle^{(i)} \to \langle B \rangle^{(i)}, \langle A \rangle^{(i)} \to X^{(i)}, X^{(i)} \to X^{(i)}$, $X^{(i)}$ being generated when the sentential form of such a grammar becomes a terminal string). Note that we need to keep track of the derivation in these grammars because it is possible that the derivation would be blocked due to the impossibility to rewrite a nonterminal sentential form of a grammar.

We still make the following observation about the derivation proccess in the grammar system: once the grammar G_1 asks for communication from a component whose sentential form is terminal, that is the end of the derivation. In other words, in any derivation proccess at most one of the grammars which is queried uses a terminal production (that is a production $A \to a, A \in N, a \in T$) before being queried.

Let $V_T = T \cup \{(0)\}$, let V_N be the set of all nonterminal symbols in the matrices that follow, let S_0 be a new symbol and let

$$G = (V_N, V_T, S_0, M),$$

be a matrix grammar, where M is the set consisting of the matrices below.

Before outlining the matrices, we mention that in all the cases we consider for any $1 \le i \le 2^n - 1$, for any $0 \le p \le n$ and for any $2 \le k_j \le n+1$, $1 \le j \le n-p$, $2 \le l_t \le n+1$, $1 \le t \le p$, that $\{k_1, \ldots, k_{n-p}, l_1, \ldots, l_p\} = \{2, 3, \ldots, n+1\}$, and when $p \ge 1$, we still consider $[l_1]^{(i,s)}[l_2]^{(i,s)} \ldots [l_p]^{(i,s)} = x_{i,s} \in \mathcal{S}_{i,p}$ for any $1 \le s \le p!$. When $p = 0$, such a string will not appear in the corresponding matrix anymore. We also assume that the matrices below are defined for any productions $r_{k_j} \in P'_{k_j}, 1 \le j \le n-p$.

To start a derivation in the matrix grammar, one should use the matrix

$$(S_0 \to S[1])$$

followed by a matrix

$$\begin{array}{lll}(S & \to & h_1(S_1)h_{l_1}(S_{l_1})h_{l_2}(S_{l_2}) \ldots h_{l_p}(S_{l_p}),\\ [1] & \to & \langle S_{k_1}\rangle^{(k_1)} \ldots \langle S_{k_{n-p}}\rangle^{(k_{n-p})}[l_1]^{(i,s)} \ldots [l_p]^{(i,s)})\end{array}$$

When simulating a derivation in the grammar system by a derivation in the matrix grammar, one has to distinguish the following situations:

(a) *to simulate a derivation step of Γ when no terminal production is used and no interrogation is performed,*

one uses a matrix like

$$\begin{array}{l}(h_1(A_1 \to a_1B_1), h_{l_1}(A_{l_1} \to a_{l_1}B_{l_1}), \ldots, h_{l_p}(A_{l_p} \to a_{l_p}B_{l_p}),\\ [l_1]^{(i,s)} \to [l_1]^{(i,s)}, \ldots, [l_p]^{(i,s)} \to [l_p]^{(i,s)}, g_{k_1}(r_{k_1}), \ldots, g_{k_{n-p}}(r_{k_{n-p}})),\end{array}$$

for any productions $A_j \to a_jB_j \in P_j, 1 \le j \le p$.

(b) *to simulate a derivation step when no terminal production is used, but a query is performed*

Note that when the grammar G_1 is querying a grammar $G_t, 2 \le t \le n+1$, it must be the case $t = l_1$, in our notations. We again distinguish several subcases:

(b.1) *Grammar G_{l_1} will not be queried anymore until the end of the derivation;*

to simulate such a case, one should use a matrix like

$$\begin{aligned}&(h_1(A_1) \to a_1, h_{l_1}(A_{l_1}) \to h_{l_1}(a_{l_1})h_1(B_{l_1}), h_{l_2}(A_{l_2} \to a_{l_2}B_{l_2}), \ldots,\\&h_{l_p}(A_{l_p} \to a_{l_p}B_{l_p}), [l_1]^{(i,s)} \to \langle S_{l_1}\rangle^{(l_1)}, [l_2]^{(i,s)} \to [l_2]^{(i',s')}, \ldots,\\&[l_p]^{(i,s)} \to [l_p]^{(i',s')}, g_{k_1}(r_{k_1}), \ldots, g_{k_{n-p}}(r_{k_{n-p}})),\end{aligned}$$

for any productions $A_1 \to a_1Q_{l_1} \in P_1$, $A_{l_j} \to a_{l_j}B_{l_j} \in P_{l_j}, 1 \le j \le p$, where $x_{i',s'} = [l_2]^{(i',s')}[l_3]^{(i',s')} \ldots [l_p]^{(i',s')} \in \mathcal{S}_{i',p-1}, M_{i'} = \{l_2, \ldots, l_p\}$.

(b.2) *if the next query of G_1 is also from the grammar G_{l_1} (and then $G_{l_2}, \ldots, G_{l_p}$ will be queried);*

one uses a matrix like

$$\begin{aligned}&(h_1(A_1) \to a_1, h_{l_1}(A_{l_1}) \to h_{l_1}(a_{l_1})h_1(B_{l_1})h_{l_1}(S_{l_1}), h_{l_2}(A_{l_2} \to a_{l_2}B_{l_2}), \ldots,\\&h_{l_p}(A_{l_p} \to a_{l_p}B_{l_p}), [l_1]^{(i,s)} \to [l_1]^{(i,s)}, [l_2]^{(i,s)} \to [l_2]^{(i,s)}, \ldots, [l_p]^{(i,s)} \to [l_p]^{(i,s)},\\&g_{k_1}(r_{k_1}), \ldots, g_{k_{n-p}}(r_{k_{n-p}})),\end{aligned}$$

for any productions $A_1 \to a_1Q_{l_1} \in P_1$, $A_j \to a_jB_j \in P_j, 1 \le j \le p$.

(b.3) *if G_{l_2} is queried next, and then G_{l_3}, maybe afterwards again G_{l_2}, and then G_{l_3}, then G_{l_4}, maybe afterwards again G_{l_2} or G_{l_3}, or G_{l_4}, and so on, until G_{l_j} is queried, and maybe afterwards one of the grammars $G_{l_2}, \ldots, G_{l_{j-1}}$, or G_{l_j} will be queried, then G_{l_1} will be queried, and then $G_{l_{j+1}}, \ldots, G_{l_p}$, maybe again $G_{l_2}, \ldots, G_{l_p}, G_{l_1}$.*

then one uses a matrix of the type

$$\begin{aligned}&(h_1(A_1) \to a_1, h_{l_1}(A_{l_1}) \to h_{l_1}(a_{l_1})h_1(B_{l_1}), h_{l_2}(A_{l_2} \to a_{l_2}B_{l_2}), \ldots,\\&h_{l_{j-1}}(A_{l_{j-1}} \to a_{l_{j-1}}B_{l_{j-1}}), h_{l_j}(A_{l_j}) \to h_{l_j}(a_{l_j}B_{l_j})h_{l_1}(S_{l_1}),\\&h_{l_{j+1}}(A_{l_{j+1}} \to a_{l_{j+1}}B_{l_{j+1}}), h_{l_p}(A_{l_p} \to a_{l_p}B_{l_p}), [l_1]^{(i,s)} \to [l_2]^{(i,s')},\\&[l_2]^{(i,s)} \to [l_3]^{(i,s')}, \ldots, [l_{j-1}]^{(i,s)} \to [l_j]^{(i,s')},\\&[l_j]^{(i,s)} \to [l_1]^{(i,s')}, [l_{j+1}]^{(i,s)} \to [l_{j+1}]^{(i,s')}, \ldots, [l_p]^{(i,s)} \to [l_p]^{(i,s')},\\&g_{k_1}(r_{k_1}), \ldots, g_{k_{n-p}}(r_{k_{n-p}})),\end{aligned}$$

for any productions $A_1 \to a_1Q_{l_1} \in P_1$, $A_{l_j} \to a_{l_j}B_{l_j} \in P_{l_j}, 1 \le j \le p$, $x_{i,s'} = [l_2]^{(i,s')}[l_3]^{(i,s')} \ldots [l_j]^{(i,s')}[l_1]^{(i,s')}[l_{j+1}]^{(i,s')} \ldots [l_p]^{(i,s')} \in \mathcal{S}_{ip}$.

To deal with the case when a terminal production (that is a production $A \to a$) is used in one of the grammars $G_i, 2 \le i \le n+1$ (note that this implies that this grammar is for sure G_{l_p}, because the derivation will end after a query of a grammar whose sentential form is a terminal string), one should use the same types of matrices as those outlined for cases **(a) - (b)**, but in which one replaces the production $h_{l_p}(A_{l_p} \to a_{l_p}B_{l_p})$ with $h_{l_p}(A_{l_p} \to a_{l_p})$, where $A_{l_p} \to a_{l_p} \in P_{l_p}$ and also one replaces the production $[l_p]^{(i,s)} \to [l_p]^{(i',s')}$ (or $[l_p]^{(i,s)} \to [l_p]^{(i,s')}$ or $[l_p]^{(i,s)} \to [l_p]^{(i,s)}$) with the production $[l_p]^{(i,s)} \to \langle l_p\rangle$.

In order to simulate the continuation of the derivation after a terminal production has been applied, one should also use the same types of matrices as those provided for cases **(a) - (b)** but in which the rule $h_{l_p}(A_{l_p}) \rightarrow h_{l_p}(a_{l_p}B_{l_p})$ is erased and the rule $[l_p]^{(i,s)} \rightarrow [l_p]^{(i',s')}$ (or $[l_p]^{(i,s)} \rightarrow [l_p]^{(i,s')}$ or $[l_p]^{(i,s)} \rightarrow [l_p]^{(i,s)}$) is replaced by the rule $\langle l_p\rangle \rightarrow \langle l_p\rangle$.

A query from G_{l_p} (which then is identical to G_{l_1}, that is we have the case $p = 1$) should be simulated in such a case by a matrix like

$$(h_1(A_1) \rightarrow a_1, \langle l_1\rangle \rightarrow (0), g_{k_1}(\overline{r_{k_1}}), \ldots, g_{k_{n-1}}(\overline{r_{k_{n-1}}})),$$

for any production $A_1 \rightarrow a_1Q_{l_1} \in P_1$, where for a production $r_{k_j} = A \rightarrow \alpha \in P'_{k_j}$ we have denoted by $\overline{r_{k_j}}$ the production $A \rightarrow (0)$.

If a query from G_{l_p} (which then again is identical to G_{l_1}, that is we have the case $p = 1$) appears in the very moment when in G_{l_1} a production $A_{l_1} \rightarrow a_{l_1}$ is used, then one should use the matrix

$$(h_1(A_1) \rightarrow a_1, h_{l_1}(A_{l_1}) \rightarrow a_{l_1}, [l_1]^{(i,s)} \rightarrow (0), g_{k_1}(\overline{r_{k_1}}), \ldots, g_{k_{n-1}}(\overline{r_{k_{n-1}}})),$$

for any production $A_1 \rightarrow a_1Q_{l_1} \in P_1, A_{l_1} \rightarrow a_{l_1} \in P_{l_1}$ and $\overline{r_{k_j}}, 1 \leq j \leq n-1$ being as before.

To simulate the case when the derivation is ended by changing the sentential form of G_1 only according to derivation rules of G_1, one should use matrices

$$(h_1(A_1 \rightarrow a_1B_1), g_2(r_2), g_3(r_3), \ldots, g_{n+1}(r_{n+1})),$$

for any production $A_1 \rightarrow a_1B_1 \in P_1$ and for any productions $r_j \in P'_j, 2 \leq j \leq n+1$,

and finally

$$(h_1(A_1 \rightarrow a_1), g_2(\overline{r_2}), g_3(\overline{r_3}), \ldots, g_{n+1}(\overline{r_{n+1}})),$$

for any production $A_1 \rightarrow a_1 \in P_1$ and for any productions $r_j \in P'_j, 2 \leq j \leq n+1$.

We then obtain

$$L(G) = L(\Gamma)(0)^n.$$

It is easy to see that G is a finite index matrix grammar (any sentential form of G has at most $2n+1$ nonterminal symbols) , without $\lambda-$ productions. But $\mathbf{L}(MAT_{fin})$ is a full AFL (see [4]), hence it is closed under arbitrary morphisms. Then $L(\Gamma) \in \mathbf{L}(MAT_{fin})$, which proves the inclusion $\mathbf{L}(CPC_*REG) \subseteq \mathbf{L}(MAT_{fin})$.

The inclusion is proper because there are linear languages, hence languages in $\mathbf{L}(MAT_{fin})$, which are not in $\mathbf{L}(CPC_*REG)$ (Theorem 7.7 in [3]). $\square$

In order to illustrate the construction in the above proof, we briefly present an example in the following.

Hence let

$$\Gamma = (N, K, T, G_1, G_2, G_3)$$

be a centralized returning parallel communicating grammar system with regular components, where

$$\begin{aligned} P_1 &= \{S_1 \to aS_1, S_1 \to aQ_2, S_1 \to aQ_3, S_2 \to aS_1, S_3 \to aS_1, \\ &\quad S_1 \to a, S_2 \to a, S_3 \to a\}, \\ P_2 &= \{S_2 \to bS_2, S_2 \to b\}, \\ P_3 &= \{S_3 \to bS_3, S_3 \to b\}. \end{aligned}$$

Then $L(\Gamma) = \{a^{k_1}b^{k_2}a^{k_3}b^{k_4}\dots a^{k_n}b^{k_{n+1}}a^{k_{n+2}} \mid k_i \geq 1$, for any $i, 1 \leq i \leq n+2$,
$k_j \leq \sum_{l=1}^{j-1} k_l$, for any $j, 2 \leq j \leq n+1\}$

According to our construction

$$\begin{aligned} \mathcal{P}(\mathcal{N}) &= \{\{2\}, \{3\}, \{2,3\}\}, \\ \mathcal{S}_{1,1} &= \{[2]^{(1,1)}\}, \\ \mathcal{S}_{2,1} &= \{[3]^{(2,1)}\}, \\ \mathcal{S}_{3,1} &= \{[2]^{(3,1)}[3]^{(3,1)}\}, \\ \mathcal{S}_{3,2} &= \{[3]^{(3,2)}[2]^{(3,2)}\}. \end{aligned}$$

For space reasons, we will not list the matrices obtained according to the proposed construction, but we will simulate several significant derivation steps instead.

The first matrix to be used is $(S_0 \to S[1])$.

If one intends to simulate a derivation when only G_3 will be queried, then one has use next the matrix which requires that only this grammar should be queried (i.e. the permutation to be chosen should be a permutation of the set 3), that is

$$(S \to S_1^{(1)}S_3^{(3)}, [1] \to \langle S_2\rangle^{(2)}[3]^{(2,1)})$$

Then the derivation proceeds using matrices

$(S_1^{(1)} \to aS_1^{(1)}, S_3^{(3)} \to bS_3^{(3)}, [3]^{(2,1)} \to [3]^{(2,1)}, \langle S_2\rangle^{(2)} \to \langle S_2\rangle^{(2)})$,
if none of the grammars G_2 or G_3 use terminal productions, or
$(S_1^{(1)} \to aS_1^{(1)}, S_3^{(3)} \to b, [3]^{(2,1)} \to \langle 3\rangle, \langle S_2\rangle^{(2)} \to \langle S_2\rangle^{(2)})$,
if G_3 uses a terminal production, or

$$(S_1^{(1)} \rightarrow aS_1^{(1)}, S_3^{(3)} \rightarrow bS_3^{(3)}, [3]^{(2,1)} \rightarrow [3]^{(2,1)}, \langle S_2\rangle^{(2)} \rightarrow X_2^{(2)}),$$
$$\text{if } \mathrm{G}_2 \text{ uses a terminal production, or}$$
$$(S_1^{(1)} \rightarrow aS_1^{(1)}, S_3^{(3)} \rightarrow b, [3]^{(2,1)} \rightarrow \langle 3\rangle, \langle S_2\rangle^{(2)} \rightarrow X_2^{(2)}),$$
$$\text{if both } \mathrm{G}_2 \text{ and } \mathrm{G}_3 \text{ use terminal productions.}$$

If one intends to simulate a derivation when first G_3 will be queried a number of times, then G_2 will be queried (and then maybe again G_2, and so on), then one has use next the matrix which requires that both grammars should be queried in order 2,3

$$(S \rightarrow S_1^{(1)}S_3^{(3)}S_2^{(2)}, [1] \rightarrow [3]^{(3,2)}[2]^{(3,2)})$$

Then the derivation proceeds using matrices

$$(S_1^{(1)} \rightarrow aS_1^{(1)}, S_3^{(3)} \rightarrow bS_3^{(3)}, S_2^{(2)} \rightarrow bS_2^{(2)}[3]^{(3,2)} \rightarrow [3]^{(3,2)}, [2]^{(3,2)} \rightarrow [2]^{(3,2)}),$$
$$\text{if none of the grammars } \mathrm{G}_2 \text{ or } \mathrm{G}_3 \text{ use terminal productions, or}$$
$$(S_1^{(1)} \rightarrow aS_1^{(1)}, S_3^{(3)} \rightarrow bS_3^{(3)}, S_2^{(2)} \rightarrow b[3]^{(2,1)} \rightarrow [3]^{(2,1)}, [2]^{(3,2)} \rightarrow \langle 2\rangle),$$
$$\text{if } \mathrm{G}_2 \text{ uses a terminal production.}$$

Note that we need to make a distinction between the permutations "23" and "32" (it is important to know whether G_2 is queried first, and then G_3, or first G_3 is queried, and then G_2), and that is why we have used those indices to identify them.

When a query from G_3 occur, this will be simulated by a matrix below (again for saving space, we point out only the case when the sentential form of G_2 is not a terminal string, and neither becomes it right at this step):

$$(S_1^{(1)} \rightarrow a, S_3^{(3)} \rightarrow bS_3^{(1)}S_3^{(3)}, S_2^{(2)} \rightarrow bS_2^{(2)}[3]^{(3,2)} \rightarrow [3]^{(3,2)}, [2]^{(3,2)} \rightarrow [2]^{(3,2)}),$$
$$\text{if the next query is also from} \mathrm{G}_3, \text{ or}$$
$$(S_1^{(1)} \rightarrow a, S_3^{(3)} \rightarrow bS_3^{(1)}, S_2^{(2)} \rightarrow bS_2^{(2)}[3]^{(3,2)} \rightarrow \langle S_3\rangle^{(3)}, [2]^{(3,2)} \rightarrow [2]^{(1,1)}),$$
$$\text{if } \mathrm{G}_2 \text{ is not queried anymore until the end of the derivation, or}$$
$$(S_1^{(1)} \rightarrow a, S_3^{(3)} \rightarrow bS_3^{(1)}, S_2^{(2)} \rightarrow bS_2^{(2)}S_3^{(3)}[3]^{(3,2)} \rightarrow [2]^{(3,1)}, [2]^{(3,2)} \rightarrow [3]^{(3,1)}),$$
$$\text{if the next query is from} \mathrm{G}_2, \text{ but, after some steps,} \mathrm{G}_3 \text{ is}$$
$$\text{queried again.}$$

We end our example at this point, with the hope that the construction used in the proof of the theorem seems more natural now than it did before reading the example. □

Note that the result in the previous theorem cannot be improved for non-centralized returning PC grammar systems with regular components. More exactly, we have as an immediate consequence of [5] (where it is proved that any one-letter matrix language is regular) and of Theorem 7.14 of [3] (which states that there are non-regular one-letter languages in $\mathbf{L}(PC_3REG)$) that

Proposition 2 $\mathbf{L}(PC_3REG) - \mathbf{L}(MAT) \neq \emptyset$.

We still make the observation that Theorem 1 is not true either in the case of centralized non-returning regular PC grammar systems. More exactly, as an immediate consequence of [5] and of Theorem 7.13 of [3] (which claims that there are non-regular one-letter languages in $\mathbf{L}(NCPC_2REG)$) we obtain

Proposition 3 $\mathbf{L}(NCPC_2REG) - \mathbf{L}(MAT) \neq \emptyset$.

In case of regular PC grammar systems with terminal derivation, we know from [3] the following relations:

Proposition 4

- $\mathbf{L}_T(CPC_nREG) \subset \mathbf{L}(CF)$, *for any* $n \geq 1$ (Corollary 7.40)*;*
- $\mathbf{L}_T(NPC_2REG) = \mathbf{L}_T(CPC_2REG)$;
- $\mathbf{L}_T(NPC_3REG) - \mathbf{L}(CF) \neq \emptyset$;
- $\mathbf{L}_T(NCPC_nREG) \subseteq \mathbf{L}(NCPC_2REG)$, *for any* $n \geq 1$ (Theorem 7.39)*.*

However, we do not have any upper bound for non-centralized systems with regular components and deriving in the terminal mode, neither in the returning, nor in the non-returning case. The next theorem provides us such an upper bound for the non-returning non-centralized case.

Theorem 2 $\mathbf{L}_T(NPC_*REG) \subseteq \mathbf{L}(MAT_{fin})$.

Proof: Let

$$\Gamma = (N, K, T, G_1, G_2, \ldots, G_n)$$

be a non-returning non-centralized regular PC grammar system, $n \geq 1$, and let the derivation mode be the terminal one.

In order to simulate a non-returning non-centralized PC grammar system by a matrix grammar, let us first notice that in the grammar system, once a query of a sentential form has been satisfied, this sentential form becomes a terminal string. This implies that any grammar can make at most one query, hence in any derivation we have a finite number of queries, at most n.

Moreover, we are interested in pointing out with accuracy only the "useful" productions applied and queries performed, that is the the productions and queries that modify the string in the language generated by the system. All the other productions/queries in a derivation are of interest only because they can lead to a situation when a grammar's sentential form is not a terminal string, but it cannot be rewritten anymore, which means the blocking of the derivation. Note that from the point of view of the "useful" queries, a derivation either has no such queries, that is the final string is modified only according to productions of the first component, or it has, and then G_1 is the last grammar which queries another grammar.

In other words, in any derivation one can point out a chain of grammars

$$G_{l_1}, G_{l_2}, \ldots, G_{l_p},\ p \geq 1, 2 \leq l_i \leq n, 2 \leq i \leq p,$$

such that $l_1 = 1$ and in the derivation it is the case that at a given moment $G_1(= G_{l_1})$ asks for communication from grammar G_{l_2}, which has asked at a previous derivation step (or it is just asking) for communication from grammar G_{l_3}, which in turn has asked for communication (or it is just asking) from another grammar, G_{l_3}, and so on, the first query performed in the derivation being of grammar $G_{l_{p-1}}$, which queries G_{l_p}. Moreover, note that because circular queries are not allowed and because the sentential form of a grammar cannot be modified anymore after it has been queried or it has been satisfied by its query, in the chain of grammars

$$G_{l_1}, G_{l_2}, \ldots, G_{l_p},$$

we must have only grammars which are not identical with each others. But it can be the case that a grammar not in this chain asks for communication from a grammar in the chain, and still it can be the case that grammars not in that chain ask for communication from each other.

With these observations, let us note that we can simulate a derivation in the non-returning non-centralized grammar system, with regular components and deriving in a terminal mode, by a derivation in a matrix grammar in a similar way as in Theorem 1.

Then let

$$\mathcal{N} = \{[1], [2], \ldots, [n]\}$$

and let $\mathcal{P}(\mathcal{N})$ be the power set of $\mathcal{N}$ minus the empty set. Let us consider $\mathcal{N} = \{M_1, M_2, \ldots, M_{2^n-1}\}$.

Also, let $\mathcal{S}_{i,q_i}, x_{i,j}$ be just as in Theorem 1, and also let the morphisms $h_i, g_i, 1 \leq i \leq n$ be defined just as in that theorem, but, in addition, with

$$h_i(A \rightarrow aQ_j) = A^{(i)} \rightarrow a,$$

$$(\text{this is actually a short cut from } h_i(Q_j) = \lambda)$$
$$g_i(A \to aQ_j) = \langle A \rangle^{(i)} \to X^{(i)},$$

for any $2 \leq j \leq n, A \to aQ_j \in P_i$.

We will simulate the derivation in the grammar system Γ by a derivation in a matrix grammar whose sentential forms are of the type

$$h_{l_1}(\alpha_{l_1})h_{l_2}(\alpha_{l_3}) \dots h_{l_p}(\alpha_{l_p})\beta x, p \geq 1,$$

where $l_1 = 1$ and $x = b_{l_p} b_{l_{p-1}} \dots b_{l_1}, b_{l_j} \in \{[a_{l_j}]^{(i)}, \langle l_j \rangle\}$, such that $x_{i,s} = [l_p]^{(i,s)}[l_{p-1}]^{(i,s)} \dots [l_1]^{(i,s)} \in \mathcal{S}_{i,p}$ for that i for which $M_i = \{[l_1], \dots, [l_p]\} \in \mathcal{P}(\mathcal{N})$ and $\beta \in \{h_i(X), h_i(A) | A \in N, 2 \leq i \leq n\}^*, |\beta| = n - p$ and even more, $\{l_1, l_2, \dots, l_p\} \cap \{i \mid$ there exists a symbol superscribed with (i) in $\beta, 2 \leq i \leq n\} = \emptyset$.

The meaning of the symbols in this sentential form is just as in Theorem 1.

Then let $V_T = T \cup \{(0)\}$, let V_N be the set of all nonterminal symbols in the matrices that follow, let S_0 be a new symbol and let

$$G = (V_N, V_T, S_0, M),$$

be a matrix grammar, where M is the set consisting of the matrices outlined below.

We make similar conventions as in Theorem 1 concerning the notations, namely we mention that in all the matrices below we consider for any $1 \leq i \leq 2^n - 1$, for any $1 \leq p \leq n$ and for any $2 \leq k_j \leq n$, $1 \leq j \leq n - p$, $1 \leq l_t \leq n$, $1 \leq t \leq p$, that $\{k_1, \dots, k_{n-p}, l_1, \dots, l_p\} = \{2, 3, \dots, n+1\}$, $[l_1]^{(i,s)}[l_2]^{(i,s)} \dots [l_p]^{(i,s)} = x_{i,s} \in \mathcal{S}_{i,p}$ for any $1 \leq s \leq p!$. We also consider that the matrices below are defined for any productions $r_{k_j} \in P_{k_j} \cup \{X^{(i)} \to X^{(i)}\}, 1 \leq j \leq n - p$. In addition, in all the cases we consider $l_1 = 1$.

Just as in the above mentioned theorem, the derivation in the matrix grammar should start by using the matrix

$$(S_0 \to S[1])$$

followed by a matrix

$$\begin{array}{rcl}(S & \to & h_{l_1}(S_{l_1})h_{l_2}(S_{l_2}) \dots h_{l_p}(S_{l_p}), \\ {[1]} & \to & \langle S_{k_1} \rangle^{(k_1)} \dots \langle S_{k_{n-p}} \rangle^{(k_{n-p})}[l_1]^{(i,s)} \dots [l_p]^{(i,s)})\end{array}$$

When performing a derivation in the matrix grammar, we again distinguish several situations:

(a) *to simulate a derivation step of* Γ *when no terminal production is used in any of the grammars* $G_{l_j}, 1 \leq j \leq p$, *and no interrogation is performed,*

one uses a matrix like

$$(h_{l_1}(A_{l_1} \to a_{l_1} B_{l_1}), h_{l_2}(A_{l_2} \to a_{l_2} B_{l_2}), \ldots, h_{l_p}(A_{l_p} \to a_{l_p} B_{l_p}),$$
$$[l_p]^{(i,s)} \to [l_p]^{(i,s)}, \ldots, [l_1]^{(i,s)} \to [l_1]^{(i,s)},$$
$$g_{k_1}(r_{k_1}), \ldots, g_{k_{n-p}}(r_{k_{n-p}})),$$

for any productions $A_j \to a_j B_j \in P_j, 1 \leq j \leq p$.

(b) *to simulate a derivation step of* Γ *when a terminal production is used in the grammar* G_{l_p} *(note that, in our notations, this is the only grammar out of* $G_{l_1}, G_{l_2}, \ldots, G_{l_p}$ *in which one can apply a terminal production) and no interrogation is performed,*

one uses matrices like

$$(h_{l_1}(A_{l_1} \to a_{l_1} B_{l_1}), \ldots, h_{l_{p-1}}(A_{l_{p-1}} \to a_{l_{p-1}} B_{l_{p-1}}), h_{l_p}(A_{l_p} \to a_{l_p}),$$
$$[l_p]^{(i,s)} \to \langle l_p \rangle, \ldots, [l_{p-1}]^{(i,s)} \to [l_{p-1}]^{(i,s)}, \ldots, [l_1]^{(i,s)} \to [l_1]^{(i,s)},$$
$$g_{k_1}(r_{k_1}), \ldots, g_{k_{n-p}}(r_{k_{n-p}})),$$

for any productions $A_j \to a_j B_j \in P_j, 1 \leq j \leq p-1, A_p \to a_p \in P_p$.

(c) *to simulate a derivation step of* Γ *when no terminal production is used in any of the grammars* $G_{l_j}, 1 \leq j \leq p$, *but interrogations are performed by some of these grammars, namely for a* t, $0 \leq t \leq p-1$ *and a* j, $2 \leq j \leq p-t-1$, *we have the case that* G_{l_j} *asks* $G_{l_{j+1}}$ *for communication, and, in turn,* $G_{l_{j+1}}$ *queries* $G_{l_{j+2}}$, *and so on, and* $G_{l_{p-t-1}}$ *queries* $G_{l_{p-t}}$, *which sentential form is already a terminal string*

one uses matrices like

$$(h_{l_1}(A_{l_1} \to a_{l_1} B_{l_1}), \ldots, h_{l_{j-1}}(A_{l_{j-1}} \to a_{l_{j-1}} B_{l_{j-1}}), h_{l_j}(A_{l_j} \to a_{l_j} Q_{l_{j+1}}),$$
$$h_{l_{j+1}}(A_{l_{j+1}} \to a_{l_{j+1}} Q_{l_{j+2}}), \ldots, h_{l_{p-t-1}}(A_{l_{p-t-1}} \to a_{l_{p-t-1}} Q_{l_{p-t}}),$$
$$\langle l_p \rangle \to \langle l_p \rangle, \ldots, \langle l_{p-t} \rangle \to \langle l_{p-t} \rangle, [l_{p-t-1}]^{(i,s)} \to \langle l_{p-t-1} \rangle, \ldots, [l_j]^{(i,s)} \to \langle l_j \rangle,$$
$$[l_{j-1}]^{(i,s)} \to [l_{j-1}]^{(i,s)}, \ldots, [l_1]^{(i,s)} \to [l_1]^{(i,s)}, g_{k_1}(r_{k_1}), \ldots, g_{k_{n-p}}(r_{k_{n-p}})),$$

for any productions $A_{l_j} \to a_{l_j} B_{l_j} \in P_{l_j}, 1 \leq j \leq j-1, A_{l_k} \to a_{l_k} Q_{l_{k+1}} \in P_{l_k}, j-1 \leq k \leq p-t-1$.

(d) *to simulate a derivation step of* Γ *when a terminal production is used in grammar* G_{l_p}, *and interrogations are performed, by some of the grammars* $G_{l_j}, 1 \leq j \leq p-1$, *namely for a* j, $2 \leq j \leq p-1$, *we have the case that* G_{l_j} *asks* $G_{l_{j+1}}$ *for communication, and, in turn,* $G_{l_{j+1}}$ *queries* $G_{l_{j+2}}$, *and so on, and* $G_{l_{p-1}}$ *queries* G_{l_p}, *which now uses a terminal production,*

one uses matrices like

$$(h_{l_1}(A_{l_1} \to a_{l_1}B_{l_1}), \ldots, h_{l_{j-1}}(A_{l_{j-1}} \to a_{l_{j-1}}B_{l_{j-1}}), h_{l_j}(A_{l_j} \to a_{l_j}Q_{l_{j+1}}),$$
$$h_{l_{p-1}}(A_{l_{p-1}} \to a_{l_{p-1}}Q_{l_p}), h_{l_p}(A_{l_p} \to a_{l_p}), [l_p]^{(i,s)} \to \langle l_p\rangle, \ldots, [l_j]^{(i,s)} \to \langle l_j\rangle,$$
$$[l_{j-1}]^{(i,s)} \to [l_{j-1}]^{(i,s)}, [l_1]^{(i,s)} \to [l_1]^{(i,s)}, g_{k_1}(r_{k_1}), \ldots, g_{k_{n-p}}(r_{k_{n-p}})),$$

for any productions $A_{l_j} \to a_{l_j}B_{l_j} \in P_{l_j}, 1 \leq j \leq j-1, A_{l_k} \to a_{l_k}Q_{l_{k+1}} \in P_{l_k}, j-1 \leq k \leq p-1, A_{l_p} \to a_{l_p} \in P_{l_p}$.

The step when G_1 (in our notations G_{l_1} is asking for communications should be simulated with matrices like

$$(h_{l_1}(A_{l_1} \to a_{l_1}Q_{l_2}), h_{l_2}(A_{l_2} \to a_{l_2}Q_{l_3}), \ldots, h_{l_{p-t-1}}(A_{l_{p-t-1}} \to a_{l_{p-t-1}}Q_{l_{p-t}}),$$
$$\langle l_p\rangle \to (0), \ldots, \langle l_{p-t}\rangle \to (0), [l_{p-t-1}]^{(i,s)} \to (0), \ldots, [l_1]^{(i,s)} \to (0),$$
$$g_{k_1}(\overline{r_{k_1}}), \ldots, g_{k_{n-p}}(\overline{r_{k_{n-p}}})),$$

for any productions $A_{l_k} \to a_{l_k}Q_{l_{k+1}} \in P_{l_k}, 1 \leq k \leq p-t-1$, where for a production $r_{k_j} = A \to \alpha$, we have denoted $\overline{r_{k_j}} = A \to (0)$;

and still with matrices like

$$(h_{l_1}(A_{l_1} \to a_{l_1}Q_{l_2}), \ldots, h_{l_{p-1}}(A_{l_{p-1}} \to a_{l_{p-1}}Q_{l_p}), h_{l_p}(A_{l_p} \to a_{l_p}),$$
$$[l_p]^{(i,s)} \to (0), [l_{p-1}]^{(i,s)} \to (0), \ldots, [l_1]^{(i,s)} \to (0), g_{k_1}(\overline{r_{k_1}}), \ldots, g_{k_{n-p}}(\overline{r_{k_{n-p}}})),$$

for any productions $A_{l_k} \to a_{l_k}Q_{l_{k+1}} \in P_{l_k}, 1 \leq k \leq p-1, A_{l_p} \to a_{l_p} \in P_{l_p}$, where $\overline{r_k}$ are just as in the previous case.

If the final string is to be modified only according to productions of P_1, then we use matrices like:

$$(h_1(A_1 \to a_1B_1), [l_1]^{(i,s)} \to [l_1]^{(i,s)}, g_2(r_2), g_3(r_3), \ldots, g_n(r_n)),$$

and

$$(h_1(A_1 \to a_1), [l_1]^{(i,s)} \to (0), g_2(\overline{r_2}), g_3(\overline{r_3}), \ldots, g_{n+1}(\overline{r_{n+1}})),$$

for any production $A_1 \to a_1, A_1 \to a_1B_1 \in P_1$ and for any productions $r_j \in P'_j, 2 \leq j \leq n, \overline{r_j}$ being as before.

To take into account also the cases when grammars $G_{k_1}, \ldots, G_{k_{n-p}}$ (that is grammars which are not involved in the query chain including G_1) perform queries, the matrices used are very much the same as those outlined above, but in which we have to put the restrictions

- if the case is $r_{k_t} = A_{k_t} \to a_{k_t}Q_i$ and $i \in \{l_1, \ldots, l_p\}$, then we must add conditions for the sentential form of G_i to be (to become right now) a terminal one or to be in a chain of queries that are to be performed just now. This should be done in a similar manner as in the cases presented above.

- if $r_{k_h} = A \to aQ_{k_t}, 1 \leq h, t \leq n-p$, then it must be the case $r_{k_t} \in \{A_{k_t} \to a_{k_t}, X^{(k_t)} \to X^{(k_t)}, A_{k_t} \to a_{k_t}Q_i\}$;

We then obtain

$$L(G) = L(\Gamma)(0)^n.$$

But one can see that G is a finite index matrix grammar (any sentential form of G has at most $2n$ nonterminal symbols), without $\lambda-$ productions. We know that $\mathbf{L}(MAT_{fin})$ is closed under arbitrary morphisms, hence $L(\Gamma) \in \mathbf{L}(MAT_{fin})$, which proves the theorem. $\square$

4 Conclusions

We have investigated relations about regular parallel communicating grammar systems. By proving that some classes of such systems can be simulated by finite index matrix grammars, we have proved that they are not very powerful, but still powerful enough (they can generate non context-free languages). And as an immediate consequence of the results outlined in this paper and of the results in [7] we have that regular parallel communicating grammar systems are in many (actually in the most of the) cases less powerful than context-free parallel communicating grammar systems.

Many topics still remain to be investigated: there is no upper bound among grammars with regulated rewritings for returning non-centralized parallel communicating grammar systems with regular components, neither in the case when in the system one considers the usual derivation mode, nor in the case when one considers the terminal derivation mode.

References

[1] Atanasiu, A. – Mitrana, V.: The Modular Grammars. Internat. J. Comp. Math. Vol. 30, 1989, pp. 101-122

[2] Csuhaj-Varju, E. – Dassow, J.: On Cooperating Distributed Grammar Systems. J. Inf. Processing and Cybern. EIK. Vol. 26, 1990, pp. 49-63

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

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

[5] Hauschildt, D. – Jantzen, M.: Petri Net Algorithms in the Theory of Matrix Grammars. Acta Informatica. Vol. 31, 1994, pp. 719-728

[6] Meersman, R. – Rozenberg, G.: Cooperating Grammar Systems. In: Proc. MFCS'78 Symp. LNCS 64. Springer-Verlag, Berlin, 1978, pp. 364-374

[7] Mihalache, V.: Matrix Grammars versus Parallel Communicating Grammar Systems. In: Păun, Gh. (Ed.): Mathematical Aspects of Natural and Formal Languages. World Scientific, 1994, pp. 293-318

[8] Păun, Gh. – Sântean, L.: Parallel Communicating Grammar Systems: the Regular Case. Ann. Univ. Buc., Ser. Matem.-Inform. Vol. 38, 1989, pp. 55-63

[9] Salomaa, A.: Formal Languages. Academic Press, New York, London, 1973

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**