ELSEVIER

Theoretical Computer Science 276 (2002) 205–219

Theoretical Computer Science

www.elsevier.com/locate/tcs

Parallel communicating grammar systems with bounded resources ☆

Erzsébet Csuhaj-Varjú *, György Vaszil

Computer and Automation Research Institute, Hungarian Academy of Sciences, Kende utca 13-17, H-1111 Budapest, Hungary

Received June 2000; received in revised form January 2001; accepted February 2001
Communicated by A. Salomaa

Abstract

In this paper we study size properties of context-free returning parallel communicating grammar systems (PC grammar systems). We show that for each context-free returning PC grammar system an equivalent system of this type can be constructed, where the total number of symbols used for describing a component can be bounded by a reasonably small constant. Since context-free returning PC grammar systems determine the class of recursively enumerable languages, the result also demonstrates that the recursively enumerable language class can be economically described in terms of parallel communicating grammar systems.

Keywords: Grammar systems; Size complexity; Recursively enumerable languages

1. Introduction

Parallel communicating (PC) grammar systems are models of parallel and distributed computation in terms of formal grammars and languages [8]. These systems realize a network architecture of grammars working in a synchronized manner and communicating with each other by dynamically emerging requests. In more details, in a PC grammar system several grammars derive their own sentential forms in parallel and their work is organized in a communicating system to generate a single language. During the work of the system each component executes one rewriting step in each

☆ Research supported in part by the Hungarian Scientific Research Fund "OTKA" Grant no. T 029615.
* Corresponding author.
E-mail addresses: csuhaj@sztaki.hu (E. Csuhaj-Varjú), vaszil@sztaki.hu (Gy. Vaszil).

0304-3975/02/$ - see front matter
PII: S0304-3975(01)00172-4

time unit, and communication is done by so-called query symbols, one different symbol referring to each component of the system. When a component introduces a query symbol in its sentential form, the rewriting process stops and one or more communication steps are performed by substituting all occurrences of the query symbols with the current sentential forms of the queried component grammars, supposing that the requested string is query free. When no more query symbol is present in any of the sentential forms, the rewriting process starts again. In the so-called returning systems after communicating its current sentential form the component returns to its start symbol and begins to generate a new string. In non-returning systems the components continue the rewriting of their current sentential forms. Rewriting steps and communication determine a computation. The language defined by the system is the set of terminal words obtained as sentential forms of a dedicated component grammar, the master, during the computations which start from the initial configuration of the system.

PC grammar systems have been intensively investigated, the interested reader is referred to [2, 5, 7] for a summary of results and open problems.

Important aspects of studying PC grammar systems are how concise descriptions can be given for different language classes in term of these generative mechanisms, that is, what can we say about size properties of PC grammar systems which are able to generate languages of a certain language class. Natural size measures of PC grammar systems are the number of components, the number of non-terminals, the number of productions of the system, and the number of symbols which are necessary to describe the PC grammar system. Similarly, the number of non-terminals, the number of productions of the individual components, and the number of necessary symbols to present a component grammar belong to the basic size properties of the system.

The results obtained so far demonstrate that PC grammar systems are not only powerful but also economical computational tools. For example, in [3] it is shown that any recursively enumerable language can be generated by a context-free returning PC grammar system with not more than 11 components, and [1] proves that every recursively enumerable language can be generated by a context-free returning PC grammar system, where the number of non-terminals of the system is limited by a constant. In [4, 6] it is shown that for context-free returning PC grammar systems and for both linear returning and linear non-returning PC grammar systems equivalent systems of the same type can be constructed, where the length of the right-hand side of any production in the system is at most two. Analogous result is proved in [11] for parallel communicating $E0L$ systems and $ET0L$ systems.

In this paper we continue this line of investigations. Namely, we prove that for each context-free returning PC grammar system an equivalent system of this type can be constructed, where the total number of symbols used for describing a component is bounded by a reasonably small constant, namely, 22. Moreover, any component of the new system has at most seven productions, and there are not more than eight non-terminals different from a query symbol which occur in the production set of a component. Since context-free returning PC grammar systems generate the class of recursively enumerable languages, the result also demonstrates that the recursively

enumerable language class can be described in an economical manner by context-free PC grammar systems with bounded resources. It is an open question whether the above constants are sharp lower bounds or not.

2. Preliminaries

In this section we briefly recall some basic notions concerning parallel communicating grammar systems which are necessary to follow the paper. We assume that the reader is familiar with formal language theory; here we only specify some notations. For further details consult [9, 10].

We denote the set of all non-empty words over an alphabet V by V^+, if the empty word, λ, is included, then we use notation V^*. The number of elements of a finite set X is denoted by $card(X)$; $lg(w)$ and $|w|_X$, $w \in V^*$, $X \subseteq V$, denote the length of a word w and the number of occurrences of symbols from set X in w, respectively.

The family of context-free grammars and languages are denoted by CF and $\mathscr{L}(CF)$, the class of recursively enumerable languages is denoted by $\mathscr{L}(RE)$.

Now we recall the notion of a PC grammar system, introduced in [8].

Definition 2.1. A *parallel communicating (PC) grammar system* of degree n is an $(n+3)$-tuple $\Gamma = (N, K, T, G_1, \ldots, G_n)$, where N is a *non-terminal alphabet*, T is a *terminal alphabet*, and $K = \{Q_1, Q_2, \ldots, Q_n\}$ is an alphabet of *query symbols*. N, T, and K are pairwise disjoint sets. $G_i = (N \cup K, T, P_i, S_i)$, $1 \leqslant i \leqslant n$, called a *component* of Γ, is a usual Chomsky grammar with non-terminal alphabet $N \cup K$, terminal alphabet T, set of rewriting rules P_i, and *axiom* (or start symbol) S_i. G_1 is said to be the *master grammar* (or master) of Γ.

An n-tuple $(x_1, \ldots, x_n)$, where $x_i \in (N \cup T \cup K)^*$, $1 \leqslant i \leqslant n$, is called a *configuration* of Γ. $(S_1, \ldots, S_n)$ is said to be the *initial configuration*.

PC grammar systems change their configurations by direct derivation steps.

Definition 2.2. Let $\Gamma = (N, K, T, G_1, \ldots, G_n)$, $n \geqslant 1$, be a parallel communicating grammar system and let $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ be two configurations of Γ.

We say that $(x_1, \ldots, x_n)$ *directly derives* $(y_1, \ldots, y_n)$, denoted by $(x_1, \ldots, x_n) \Rightarrow (y_1, \ldots, y_n)$, if one of the following two cases holds:

1. There is no x_i which has an occurrence of any query symbol, that is, $x_i \in (N \cup T)^*$ for $1 \leqslant i \leqslant n$. Then for each i, $1 \leqslant i \leqslant n$, $x_i \Rightarrow_{G_i} y_i$ for $x_i \notin T^*$ (y_i is obtained from x_i by a direct derivation step in G_i) and $x_i = y_i$ for $x_i \in T^*$.

2. There is some x_i, $1 \leqslant i \leqslant n$, which contains at least one occurrence of a query symbol.

Then for each x_i with $|x_i|_K \neq 0$, $1 \leqslant i \leqslant n$, we write $x_i = z_1 Q_{i_1} z_2 Q_{i_2} \ldots z_t Q_{i_t} z_{t+1}$, where $z_j \in (N \cup T)^*$, $1 \leqslant j \leqslant t+1$, and $Q_{i_l} \in K$, $1 \leqslant l \leqslant t$. If $|x_{i_l}|_K = 0$ for each l, $1 \leqslant l \leqslant t$, then $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$ and in *returning* systems $y_{i_l} = S_{i_l}$, while in *non-returning*

systems $y_{i_l} = x_{i_l}$, $1 \leqslant l \leqslant t$. If $|x_{i_l}|_K \neq 0$ for some l, $1 \leqslant l \leqslant t$, then $y_i = x_i$. For all j, $1 \leqslant j \leqslant n$, for which y_j is not specified above, $y_j = x_j$.

The transitive and reflexive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$.

The first case is the description of a rewriting step. If no query symbol is present in any of the sentential forms, then each grammar uses one of its rewriting rules except those which have already produced a terminal string. The derivation gets blocked if a sentential form is not a terminal string but no rule can be applied to it.

The second case describes communication: if a query symbol Q_j appears in a sentential form x_i, $1 \leqslant i, j \leqslant n$, then the rewriting stops and communication must be performed.

Then all the query symbols $Q_{i_j}, 1 \leqslant j \leqslant t$, which appear in a sentential form x_i must be replaced by the current sentential form x_{i_j} of component G_{i_j} in the same communication step provided that no x_{i_j} has any occurrence of a query symbol. If one of these sentential forms, x_{i_j}, contains a query symbol, then first x_{i_j} must be made free from the queries before changing anything in x_i. The derivation gets blocked if in the obtained configuration none of the sentential forms with an occurrence of a query symbol can be made free from the queries in the above manner, that is, if a circular query has occurred.

After communicating its sentential form to another component, the grammar can continue its own work in two ways: In *returning* systems the component must return to its axiom and begin to generate a new string. In *non-returning* systems the components do not return to their axioms, but continue the generation of their current strings.

Definition 2.3. The *language* generated by a parallel communicating grammar system $\Gamma = (N, K, T, G_1, \ldots, G_n)$ with $G_i = (N \cup K, T, P_i, S_i)$, $1 \leqslant i \leqslant n$, is

$$L(\Gamma) = \{\alpha_1 \in T^* \mid (S_1, \ldots, S_n) \Rightarrow^* (\alpha_1, \ldots, \alpha_n)\}.$$

Thus, the generated language consists of the terminal strings which appear as sentential forms of the master grammar, G_1.

Two PC grammar systems are said to be equivalent if they generate the same language.

We denote the class of returning PC grammar systems with at most n context-free components by PC_nCF, where $n \geqslant 1$. The corresponding language class generated by these systems is denoted by $\mathscr{L}(PC_nCF)$. When an arbitrary number of components is considered, we use $*$ in the subscript instead of n.

In the theory of PC grammar systems one of the most important notions is the transition. For a PC grammar system with n components, a transition describes an n-tuple of rules which are simultaneously applied by the components at a rewriting step.

Definition 2.4. Let $\Gamma = (N, K, T, G_1, \ldots, G_n)$, $n \geqslant 1$, be a parallel communicating grammar system with components $G_i = (N \cup K, T, P_i, S_i)$, $1 \leqslant i \leqslant n$. A *transition* of Γ is an

n-tuple $\bar{r}=(r_1,\ldots,r_n)$, where $r_i\in(P_i\cup\{\#\})$, $1\leqslant i\leqslant n$, and $\#$ is an additional symbol, $\#\notin(N\cup K\cup T)$.

A transition $\bar{r}=(r_1,\ldots,r_n)$ is applied in a rewriting step $(\alpha_1,\ldots,\alpha_n)\Rightarrow(\beta_1,\ldots,\beta_n)$ of Γ if $\alpha_i\Rightarrow_{G_i}\beta_i$ by applying r_i for $r_i\in P_i$ and $\beta_i=\alpha_i$, $\alpha_i\in T^*$, for $r_i=\#$, $1\leqslant i\leqslant n$.

Now we define size complexity measures for PC grammar systems. In the case of Chomsky grammars the most well-known size measures are the number of non-terminals (Var), the number of productions ($Prod$), and the total number of symbols used to describe the productions ($Symb$) (that is, for a grammar $G=(N,T,P,S)$ we define $Symb(G)=\sum_{\alpha\to\beta\in P}(lg(\alpha\beta)+1)$.

For components of PC grammar systems similar measures can be introduced.

Definition 2.5. Let $\Gamma=(N,K,T,G_1,\ldots,G_n)$ be a parallel communicating grammar system with components $G_i=(N\cup K,T,P_i,S_i)$, $1\leqslant i\leqslant n$. We define

$$CVar(\Gamma)=\max\{card(N_i)\mid N_i \text{ is the set of non-terminals of } \Gamma \text{ which appear in } P_i, 1\leqslant i\leqslant n\},$$

$$CProd(\Gamma)=\max\{card(P_i)\mid 1\leqslant i\leqslant n\}, \quad \text{and}$$

$$CSymb(\Gamma)=\max\{Symb(G_i)\mid 1\leqslant i\leqslant n\}.$$

These measures give information on the size of the component grammars in the system: $CVar$ refers to the maximum of the number of non-terminals (without query symbols) which appear in the production set of a component grammar, $CProd$ denotes the maximum of the number of productions of the components. $CSymb$ describes how concise manner the components are presented.

3. Components with bounded size

In the following we shall study size parameters of components of context-free returning PC grammar systems. We show that if a language can be generated by a context-free returning PC grammar system, then it can be obtained by a PC grammar system presented economically as well; namely, for each returning PC grammar system with context-free components an equivalent one can be constructed with component grammars having a limited number of productions and non-terminals. Moreover, the productions of the constructed grammar system have at most two symbols at their right-hand sides. The components of the system are described in a concise manner: the total number of symbols used to describe the grammar, size measure $CSymb$, is limited by a reasonably small constant. Since context-free returning PC grammar systems generate the class of recursively enumerable languages, these results prove the existence of economical presentations of this language class in terms of PC grammar systems.

We first show that each language generated by a context-free returning PC grammar system can also be generated by a PC grammar system of the same type having *CProd*-complexity bounded with a small constant.

Theorem 3.1. *For each context-free returning PC grammar system Γ we can construct an equivalent context-free returning PC grammar system Γ' such that $CProd(\Gamma') \leqslant 7$.*

Proof. Let $\Gamma = (N, K, T, G_1, \ldots, G_n)$ be a context-free returning PC grammar system with components $G_i = (N \cup K, T, P_i, S_i)$, $1 \leqslant i \leqslant n$, and let $R = \{\bar{r}_1, \ldots, \bar{r}_t\}$, $t \geqslant 1$, be the set of its transitions. We shall define a PC grammar system $\Gamma' = (N', K', T, G'_1, \ldots, G'_m)$, $m \geqslant 1$, where each component has at most seven productions and PC grammar systems Γ and Γ' generate the same language. Γ' is constructed consisting of t collections of grammars, U_k, $1 \leqslant k \leqslant t$, where $t = card(R)$, and each collection is responsible for reproducing the effect of executing/skipping the application of transition $\bar{r}_k$, $1 \leqslant k \leqslant t$.

The idea behind the construction is that each derivation in Γ corresponds to a sequence of computation steps $c_1, c_2, \ldots, c_l$, $l \geqslant 1$, where each c_i, $1 \leqslant i \leqslant l$, corresponds either to the application of transition $\bar{r}_j$, or the skipping of the application of transition $\bar{r}_j$ where $i = kt + j$, $k \geqslant 0$, $1 \leqslant j \leqslant t$. For example, a derivation where transitions $\bar{r}_1$ and $\bar{r}_3$ are applied in succession corresponds to computational steps c_1, c_2, c_3, where c_1 stands for the application of $\bar{r}_1$, c_2 for the skipping of $\bar{r}_2$, and c_3 stands for the application of $\bar{r}_3$. When simulating a derivation of Γ corresponding to $c_1, c_2, \ldots, c_l$, where $l = kt + j$, $k \geqslant 0$, $1 \leqslant j \leqslant t$, the tuples of sentential forms "travel" through the grammar collections of Γ' from U_1 to U_t k times, and then from U_1 to U_j, and the action performed by collection U_s with $i = kt + s$, $1 \leqslant i \leqslant l$, corresponds to either the application or the skipping of transition $\bar{r}_s$ prescribed by c_i.

To help the reader understand how these grammar collections of Γ' are constructed, we briefly describe the collection U_k assigned to the kth transition, $\bar{r}_k = (r_{k,1}, \ldots, r_{k,n})$, $1 \leqslant k \leqslant t$. First, it has an n-tuple of grammars for storing the sentential forms on which the action of the collection will be performed, and a component which selects this action by introducing non-terminal E (*for execute*) or F (*for forward*). If non-terminal E is selected, then the stored sentential forms are communicated to two n-tuples of grammars; one of them checks whether the rules (or #) of $\bar{r}_k$ can be applied and the other one applies the rules simulating this way the application of $\bar{r}_k$. If non-terminal F is selected, then the stored sentential forms are communicated to an n-tuple of components which leave them unchanged, simulating this way the skipping of $\bar{r}_k$. After this simulating process, the sentential forms are communicated to the storing components of the next collection. In addition, each collection has several assistant components which will be described later.

Before formally defining Γ', we observe the following. Without loss of generality, we can assume that in any communication process occurring during any derivation of Γ, each component sends its sentential form in at most one of the communication steps. If this assumption does not hold, we can modify the transitions in R obtaining

a new set of transitions such that the results of the communications induced by this new set of transitions are not changed but the communications have the above required property. The modification is as follows: Those transitions which have no rule with an occurrence of some query symbol remain unchanged. If some production of a transition $\bar{r}_k$, $1 \leqslant k \leqslant t$, has at least one query symbol at its right-hand side, then we replace $\bar{r}_k$ by another transition $\bar{r}_k'$ such that at the end of the communication sequence following the application of $\bar{r}_k'$ the obtained configuration will be the same as the configuration we would obtain at the end of the communication sequence following the application of $\bar{r}_k$, and during the communication sequence following the application of $\bar{r}_k'$ every queried component communicates its string in exactly one of the communication steps. To see how this can be achieved, let us consider a communication sequence where there is a component, G_j, $1 \leqslant j \leqslant n$, which communicates its sentential form in more than one communication steps during the communication sequence. Then G_j sends S_j, its axiom, to all querying components which did not receive its sentential form for the first time, because it had already returned to the axiom. Since we can establish which occurrences of Q_j will be replaced by S_j in the sentential forms, we can modify the rules $X \to \alpha Q_j \beta$ in $\bar{r}_k$ by changing these occurrences of Q_j for S_j. If we make these modifications for each query symbol with the above property, we obtain a new transition having the same effect as the original one and having the required property; namely, each communicating component is active in exactly one of the communication steps of the communication process.

Now we define Γ' with grammar collections constructed to transitions $\bar{r}_k$ of R, $1 \leqslant k \leqslant t$, having the property described above.

$$
\begin{aligned}
\Gamma' = (N', K', T, G_1^M,\ & \\
& G_{1,1}^{\mathrm{st}}, \ldots, G_{1,n}^{\mathrm{st}}, G_1^{\mathrm{sel}}, G_1^{\mathrm{ini}}, G_{1,1}^{\mathrm{prep}}, \ldots, G_{1,n}^{\mathrm{prep}}, \\
& G_{1,1}^{\mathrm{ch}}, \ldots, G_{1,n}^{\mathrm{ch}}, G_{1,1}^{\mathrm{exe}}, \ldots, G_{1,n}^{\mathrm{exe}}, G_{1,1}^{\mathrm{fw}}, \ldots, G_{1,n}^{\mathrm{fw}}, \\
& G_{1,1}^{\mathrm{clean}}, \ldots, G_{1,n}^{\mathrm{clean}}, \ldots, G_t^M, \\
& G_{t,1}^{\mathrm{st}}, \ldots, G_{t,n}^{\mathrm{st}}, G_t^{\mathrm{sel}}, G_t^{\mathrm{ini}}, G_{t,1}^{\mathrm{prep}}, \ldots, G_{t,n}^{\mathrm{prep}}, \\
& G_{t,1}^{\mathrm{ch}}, \ldots, G_{t,n}^{\mathrm{ch}}, G_{t,1}^{\mathrm{exe}}, \ldots, G_{t,n}^{\mathrm{exe}}, G_{t,1}^{\mathrm{fw}}, \ldots, G_{t,n}^{\mathrm{fw}}, \\
& G_{t,1}^{\mathrm{clean}}, \ldots, G_{t,n}^{\mathrm{clean}}),
\end{aligned}
$$

where $t = card(R)$. Let G_1^M be the master grammar. Let

$$
\begin{aligned}
N' = & \{S\} \cup N \cup \{[A] \mid A \in N\} \\
& \cup \{E, F, E', F', E'', F'', \tilde{E}, \tilde{F}\} \\
& \cup \{D, D', S^1, S^2, S^3, S'\}
\end{aligned}
$$

and let S be the start symbol of each component grammar of Γ'.

Let us define the production sets of the components as follows: For k, $1 \leqslant k \leqslant t-1$, let

$$P_k^M = \{S \to S^1, S^1 \to S^2, S^2 \to S^3, S^3 \to Q_{k,1}^{\text{exe}}\} \cup \{S^3 \to S\} \cup \{S^3 \to Q_{k+1}^M\},$$

and

$$P_t^M = \{S \to S^1, S^1 \to S^2, S^2 \to S^3, S^3 \to Q_{t,1}^{\text{exe}}\} \cup \{S^3 \to S\}.$$

(These are the productions of the components which select the terminal words.)

Let for $1 \leqslant k \leqslant t$ and $1 \leqslant i \leqslant n$,

$$P_{k,i}^{\text{st}} = \{D \to D', D' \to \lambda, S \to S', S' \to S_i\} \cup \{S \to Q_j^{\text{ini}}, \tilde{F} \to Q_{j,i}^{\text{fw}} D\} \cup \bar{P}_{k,i}^{\text{st}},$$

where

$$\bar{P}_{k,i}^{\text{st}} = \begin{cases} \{\tilde{E} \to S_i D\} & \text{if } com(\bar{r}_j) = 1 \text{ and } ind(\bar{r}_j, i) = S_i, \\ \{\tilde{E} \to Q_{j,i}^{\text{exe}} D\} & \text{otherwise,} \end{cases}$$

and $j = k-1$ for $2 \leqslant k \leqslant t$ and $j = t$ for $k = 1$. For a transition $\bar{r}_k$ of Γ, we define $com(\bar{r}_k) = 1$ if the transition induces communication and we set $com(\bar{r}_k) = 0$, if the transition does not introduce any query symbol. For a transition $\bar{r}_k$ which induces communication, we denote by $ind(\bar{r}_k, i) = S_i$ if the communication sequence following $\bar{r}_k$ results in S_i at the ith component.

(These components store the n-tuples of sentential forms before starting the application/skipping of the transition.)

For each k, $1 \leqslant k \leqslant t$, let

$$P_k^{\text{sel}} = \{S \to E, S \to F, E \to E', F \to F', E' \to S, F' \to S\}.$$

(This component decides whether the kth transition will be executed or its application will be skipped.)

Let for each k, $1 \leqslant k \leqslant t$,

$$P_k^{\text{ini}} = \{S \to Q_k^{\text{sel}}, E \to E', F \to F', E' \to \tilde{E}, F' \to \tilde{F}\}.$$

(This component stores the information whether the kth transition was selected to be executed or skipped.)

Let for $1 \leqslant k \leqslant t$ and $1 \leqslant i \leqslant n$,

$$P_{k,i}^{\text{prep}} = \{S \to Q_k^{\text{sel}}, E \to Q_{k,i}^{\text{st}}, F \to F', F' \to F'', F'' \to S\} \cup \{X \to [X] \mid r_{k,i} = X \to \alpha\}.$$

(These components perform a preparatory step on the n-tuple of sentential forms before executing transition $\bar{r}_k = (r_{k,1}, \ldots, r_{k,n})$).

Let for $1 \leqslant k \leqslant t$ and $1 \leqslant i \leqslant n$,

$$P_{k,i}^{\mathrm{ch}} = \{S \to Q_k^{\mathrm{sel}}, F \to F', F' \to F'', F'' \to \tilde{F}\} \cup \bar{P}_{k,i}^{\mathrm{ch}},$$

where

$$\bar{P}_{k,i}^{\mathrm{ch}} = \begin{cases} \{E \to Q_{k,i}^{\mathrm{st}}\} & \text{for } r_{k,i} = \#, \\ \{E \to Q_{k,i}^{\mathrm{st}} D, X \to X\} & \text{for } r_{k,i} = X \to \alpha. \end{cases}$$

(These components check whether or not the rules of transition $\bar{r}_k = (r_{k,1}, \ldots, r_{k,n})$ can be applied to the considered n-tuple of sentential forms.)

Let for $1 \leqslant k \leqslant t$ and $1 \leqslant i \leqslant n$,

$$P_{k,i}^{\mathrm{exe}} = \{S \to Q_k^{\mathrm{sel}}, E \to E', E' \to Q_{k,i}^{\mathrm{prep}}, F \to F', F' \to F'', F'' \to S\} \cup \{[X] \to \langle \alpha^{(k,i)} \rangle \mid r_{k,i} = X \to \alpha\},$$

where $\langle \alpha^{(k,i)} \rangle$ is defined as follows: if $\alpha = z_1 Q_{i_1} z_2 \ldots z_s Q_{i_s} z_{s+1}$, where $z_j \in (N \cup T)^*$, $1 \leqslant j \leqslant s+1$, $i_j \in \{1, \ldots, n\}$. Then $\langle \alpha^{(k,i)} \rangle = z_1 Q_{k,i_1}^{\mathrm{exe}} z_2 \ldots z_s Q_{k,i_s}^{\mathrm{exe}} z_{s+1}$.

(These components simulate the execution of $r_{k,i}$ from $\bar{r}_k$.)

Let for $1 \leqslant k \leqslant t$ and $1 \leqslant i \leqslant n$,

$$P_{k,i}^{\mathrm{fw}} = \{S \to Q_k^{\mathrm{sel}}, E \to E', E' \to E'', E'' \to S, F \to Q_{k,i}^{\mathrm{st}} D, D \to D', D' \to \lambda\}.$$

(These components store the n-tuple of sentential forms before forwarding them to the next grammar collection.)

Finally, let for $1 \leqslant k \leqslant t$ and $1 \leqslant i \leqslant n$,

$$P_{k,i}^{\mathrm{clean}} = \{S \to S^1, S^1 \to S^2, S^2 \to S^3, S^3 \to Q_{k,i}^{\mathrm{ch}} S\}.$$

(These components remove the sentential form from component $G_{k,i}^{\mathrm{ch}}$ when it is not needed anymore.)

In the following we shall prove that grammar system Γ' generates the same language as Γ by showing that the grammar collections U_k,

$$(\ldots, G_k^M, G_{k,1}^{\mathrm{st}}, \ldots, G_{k,n}^{\mathrm{st}}, G_k^{\mathrm{sel}}, G_k^{\mathrm{ini}}, G_{k,1}^{\mathrm{prep}}, \ldots, G_{k,n}^{\mathrm{prep}}, G_{k,1}^{\mathrm{ch}}, \ldots, G_{k,n}^{\mathrm{ch}}, G_{k,1}^{\mathrm{exe}}, \ldots, G_{k,n}^{\mathrm{exe}}, G_{k,1}^{\mathrm{fw}}, \ldots, G_{k,n}^{\mathrm{fw}}, G_{k,1}^{\mathrm{clean}}, \ldots, G_{k,n}^{\mathrm{clean}}, \ldots)$$

of Γ' are able to simulate the application or the skipping of transition $\bar{r}_k$, $1 \leqslant k \leqslant t$.

Suppose that at some stage of a derivation in Γ', for all k, $1 \leqslant k \leqslant t$, components G_k^M have sentential forms $u_k \in \{S\} \cup T^*$; $G_{k,i}^{\mathrm{st}}$, $1 \leqslant i \leqslant n$, have sentential forms $\alpha_{k,i} D$; G_k^{sel},

G_k^{ini}, $G_{k,i}^{\mathrm{prep}}$, $G_{k,i}^{\mathrm{ch}}$, $G_{k,i}^{\mathrm{exe}}$, $G_{k,i}^{\mathrm{fw}}$, $1 \leqslant i \leqslant n$, have sentential forms S, and $G_{k,i}^{\mathrm{clean}}$, $1 \leqslant i \leqslant n$, have sentential forms $\delta_{k,i}S$, where $\delta_{k,i} \in (N' \cup T)^*$; that is, each collection is in a configuration of the form

$$(\ldots, u_k, \alpha_{k,1}D, \ldots, \alpha_{k,n}D, S, S, S, \ldots, S, S, \ldots, S,$$
$$S, \ldots, S, S, \ldots, S, \delta_1 S, \ldots, \delta_n S, \ldots).$$

We shall see that starting from a configuration of this form, the grammar collection U_k simulates the effect of the application or the skipping of transition $\bar{r}_k$ on the sentential forms $\alpha_{k,i}$, $1 \leqslant i \leqslant n$, and then sends the result to the next collection, U_r, where $r = k+1$ for $1 \leqslant k \leqslant t-1$ and $r = 1$ for $k = t$. In the following, for legibility we sometimes do not indicate the range of i, k; that is, $1 \leqslant i \leqslant n$, $1 \leqslant k \leqslant t$.

In the first rewriting step G_k^{sel} selects the activity to be performed, the execution or the skipping of transition $\bar{r}_k$, by introducing non-terminal E or F. The selected non-terminal is then communicated to G_k^{ini}, G_k^{prep}, $G_{k,i}^{\mathrm{ch}}$, $G_{k,i}^{\mathrm{exe}}$, and $G_{k,i}^{\mathrm{fw}}$, $1 \leqslant i \leqslant n$. Meanwhile, $G_{k,i}^{\mathrm{st}}$ rewrite $\alpha_{k,i}D$ to $\alpha_{k,i}D'$, and $G_{k,i}^{\mathrm{clean}}$ rewrite $\delta_{k,i}S$ to $\delta_{k,i}S^1$, $1 \leqslant i \leqslant n$.

Now, if E is selected, then sentential forms $\alpha_{k,i}D'$ are rewritten to $\alpha_{k,i}$ by $G_{k,i}^{\mathrm{st}}$ and then they are communicated to $G_{k,i}^{\mathrm{prep}}$ and $G_{k,i}^{\mathrm{ch}}$, $1 \leqslant i \leqslant n$. The other components rewrite in this step E to E', and components $G_{k,i}^{\mathrm{clean}}$ rewrite $\delta_{k,i}S^1$ to $\delta_{k,i}S^2$, $1 \leqslant i \leqslant n$. In the next step, G_k^{ini} rewrites E' to $\tilde{E}$, and then communicates this non-terminal to $G_{r,i}^{\mathrm{st}}$, r defined as above, the storing components of the next collection. At the same time, grammars $G_{k,i}^{\mathrm{st}}$ receive the sentential form of G_s^{ini} of the previous collection, where $s = k-1$ for $k \geqslant 2$, and $s = t$ for $k = 1$. In this same step, each component $G_{k,i}^{\mathrm{prep}}$ rewrites non-terminal X to $[X]$ in $\alpha_{k,i}$, where $r_{k,i} = X \to \alpha$, or leaves $\alpha_{k,i}$ unchanged if it is a terminal string. Next they send the sentential forms to $G_{k,i}^{\mathrm{exe}}$; $G_{k,i}^{\mathrm{clean}}$ rewrites S^2 to S^3; $G_{k,i}^{\mathrm{fw}}$ rewrites E' to E''; and $G_{k,i}^{\mathrm{ch}}$ checks whether or not the rule $r_{k,i}$ of transition $\bar{r}_k$ can be applied to $\alpha_{k,i}$, $1 \leqslant i \leqslant n$. This checking process is done as follows. If $r_{k,i} = \#$, which means that $\alpha_{k,i}$ should be a terminal word, then $G_{k,i}^{\mathrm{ch}}$ has received $\alpha_{k,i}$ by introducing $Q_{k,i}^{\mathrm{st}}$, so it has $\alpha_{k,i}$ as sentential form. If this sentential form is not terminal, then the system is blocked, because $G_{k,i}^{\mathrm{ch}}$ does not have any rule for any non-terminal from N. If $r_{k,i} = X \to \alpha$, then to perform the transition in a correct manner, $\alpha_{k,i}$ should contain the non-terminal $X \in N$. In this case, $G_{k,i}^{\mathrm{ch}}$ have received $\alpha_{k,i}$ by introducing $Q_{k,i}^{\mathrm{st}}D$, so now it has $\alpha_{k,i}D$ as sentential form. If $\alpha_{k,i}$ does not contain the non-terminal X, then the system is blocked, because $G_{k,i}^{\mathrm{ch}}$ has a rule $X \to X$ for X, but it does not have rules for any other non-terminal from $N \cup \{D\}$.

Now, if the above described check is successful, then components $G_{k,i}^{\mathrm{exe}}$ rewrite non-terminals $[X]$ according to the rules $r_{k,i}$ of $\bar{r}_k$, and the communication possibly induced by $\bar{r}_k$ is realized by components $G_{k,i}^{\mathrm{exe}}$, $1 \leqslant i \leqslant n$, by exchanging the requested sentential forms. Since each component sends its sentential form during the communication process in at most one of the communication steps (see the assumption about R in the first part of the proof), no start symbol $S \in N' \backslash N$ is inserted in the resulting strings. When the communication is finished, each executing component which did not return to the axiom sends its sentential form to the corresponding storing component of the next

collection. If for some j, $G^{\mathrm{exe}}_{k,j}$ has returned to the axiom, then instead of requesting its sentential form, $G^{\mathrm{st}}_{r,j}$, the next storing component, introduces S_jD, $1 \leqslant j \leqslant t$.

During this process, components $G^{\mathrm{clean}}_{k,i}$ remove the sentential forms of $G^{\mathrm{ch}}_{k,i}$ by introducing $\delta_{k,i}Q^{\mathrm{ch}}_{k,i}S$; grammars $G^{\mathrm{fw}}_{k,i}$ rewrite E'' to S; G^{ini}_{k} and $G^{\mathrm{prep}}_{k,i}$ introduce Q^{sel}_{k}, while G^{sel}_{k} rewrites E' or F' to S; and grammars $G^{\mathrm{st}}_{k,i}$ receive sentential forms $\beta_{s,i}$ from $G^{\mathrm{exe}}_{s,i}$ or $G^{\mathrm{fw}}_{s,i}$ of the previous collection U_s, s as defined above.

If non-terminal F is chosen by G^{sel}_{k} at the beginning, then $G^{\mathrm{st}}_{k,i}$, $1 \leqslant i \leqslant n$, rewrite $\alpha_{k,i}D'$ to $\alpha_{k,i}$, and communicate them to $G^{\mathrm{fw}}_{k,i}$, where $Q^{\mathrm{st}}_{k,i}D$ is present, so these components have again $\alpha_{k,i}D$ as sentential forms. Meanwhile, $G^{\mathrm{clean}}_{k,i}$, $1 \leqslant i \leqslant n$, rewrite S^1 to S^2, and the other components, G^{ini}_{k}, $G^{\mathrm{prep}}_{k,i}$, $G^{\mathrm{ch}}_{k,i}$, and $G^{\mathrm{exe}}_{k,i}$ rewrite F to F'. Then G^{ini}_{k} rewrites F' to $\tilde{F}$ and sends it to $G^{\mathrm{st}}_{r,i}$, $1 \leqslant i \leqslant n$, the storing components of the next collection, U_r, where $r = k+1$ for $1 \leqslant k \leqslant t-1$ and $r = 1$ for $k = t$, while $G^{\mathrm{st}}_{k,i}$, $1 \leqslant i \leqslant n$, receive $\tilde{E}$ or $\tilde{F}$ from G^{ini}_{s} of the previous collection, U_s, $s = k-1$ for $k \geqslant 2$ and $s = t$ for $k = 1$.

During this step, components $G^{\mathrm{fw}}_{k,i}$ rewrite $\alpha_{k,i}D$ to $\alpha_{k,i}D'$; grammars $G^{\mathrm{clean}}_{k,i}$, $1 \leqslant i \leqslant n$, rewrite S^2 to S^3; and $G^{\mathrm{prep}}_{k,i}$, $G^{\mathrm{exe}}_{k,i}$, $G^{\mathrm{ch}}_{k,i}$, $1 \leqslant i \leqslant n$ rewrite F' to F''. In the next step, components $G^{\mathrm{st}}_{k,i}$ receive the sentential forms of the previous collection; $G^{\mathrm{fw}}_{k,i}$, $1 \leqslant i \leqslant n$ rewrite $\alpha_{k,i}D'$ to $\alpha_{k,i}$ and communicate them to the storing components of the next collection; grammars $G^{\mathrm{clean}}_{k,i}$ remove the sentential forms of $G^{\mathrm{ch}}_{k,i}$; $G^{\mathrm{ini}}_{k,i}$ receives S from $G^{\mathrm{sel}}_{k,i}$, and $G^{\mathrm{exe}}_{k,i}$ rewrites F'' to $S, 1 \leqslant i \leqslant n$.

To complete the discussion, we note that in both cases, either choosing E or choosing F at the beginning, at the end of the above procedure the sentential form u_k of G^{M}_{k} will be rewritten through sentential forms S^1, S^2, and S^3 to sentential form u'_k, where $u'_k \in (N' \cup T)^*$, $1 \leqslant k \leqslant t$.

Thus, in both cases, we obtain a configuration of the above form; that is

$$(\ldots, u'_k, \beta_{s,1}D, \ldots, \beta_{s,n}D, S, S, S, \ldots, S, S, \ldots, S,$$
$$S, \ldots, S, S, \ldots, S, \delta_1 v_1 S, \ldots, \delta_n v_n S, \ldots),$$

where sentential forms $\beta_{j,i}$ are the result of the chosen activity of U_j on the sentential forms $\alpha_{j,i}$, $1 \leqslant j \leqslant t$, $1 \leqslant i \leqslant n$. The string $u'_k \in (N' \cup T)^*$, the sentential form of component G^{M}_{k}, is either u_k, S, or $\beta_{l,1}$, where $k \leqslant l \leqslant t$. If $u_k \in T^*$ and component G^{M}_{s} of the preceding collection, U_s, did not query G^{M}_{k}, then $u'_k = u_k$. Otherwise, u'_k is either S or $\beta_{l,1}$, the sentential form of the first storing component of a grammar collection U_l for some $l \geqslant k$. Notice that if $\beta_{l,1}$ is not in T^*, and the derivation has not successfully terminated yet, the derivation will abnormally terminate at the next step, since G^{M}_{k} has no production for any element of N.

We can observe that the obtained configuration is in the form like the starting one; that is, the kth collection of grammars successfully simulated the skipping of transition $\bar{r}_k$ and forwarded the n-tuple of sentential forms to the next grammar collection.

Now we show that starting from the initial configuration, the first few steps lead to a configuration of the form above. During the first rewriting step G^{sel}_{k}, $1 \leqslant k \leqslant t$ introduces the non-terminal E or F, which is then sent to all the other grammars except

to components G_k^M, $G_{k,i}^{\mathrm{st}}$, and $G_{k,i}^{\mathrm{clean}}$, $1 \leqslant i \leqslant n$, which must introduce S^1, S', and S^1, respectively, otherwise the system is blocked. In the next rewriting step, components $G_{k,i}^{\mathrm{st}}$ introduce S_i, $1 \leqslant i \leqslant n$, and if G_k^{sel} selected F, then sentential forms S_i are transferred to $G_{k,i}^{\mathrm{fw}}$, and are forwarded to the next grammar collection in the same way as described above.

If during the first rewriting step E was selected by G_k^{sel}, $1 \leqslant k \leqslant t$, then after introducing S_i, $1 \leqslant i \leqslant n$, components $G_{k,i}^{\mathrm{st}}$ communicate them to components $G_{k,i}^{\mathrm{prep}}$ and $G_{k,i}^{\mathrm{ch}}$, and then they are received by $G_{k,i}^{\mathrm{exe}}$, $1 \leqslant i \leqslant n$, in the same way as described above. If it is possible to apply the rules of transition $\bar{r}_k$ to the sentential forms S_i, $1 \leqslant i \leqslant n$, then the result of their application, $\beta_{k,i}$, will be communicated to the storing components $G_{r,i}^{\mathrm{st}}$ of the next collection, U_r, where $r = k+1$ for $1 \leqslant k \leqslant t-1$ and $r = 1$ for $k = t$.

The possible actions of G_k^M are also the same as described above, so we obtain a configuration of the desired form in both cases. If $\bar{r}_k$ cannot be used, the system blocks.

By these facts, any derivation of Γ can be simulated in Γ' in the following way. Suppose, that a word w from $L(\Gamma)$ is generated by a sequence of derivation steps determined by the applied transitions and the possibly following communication sequences t_{i_j}, $i_j \in \{1, \ldots t\}$, $1 \leqslant j \leqslant s$. Then w can be obtained in Γ' as follows: Starting from the initial configuration, grammar collection i_1 simulates the execution of the transition, while the other collections choose the skipping of their transitions. Then the obtained n-tuple of grammars is forwarded through several grammar collections to collection i_2, where this transition and the possibly following communication sequence will be applied. Meantime the other grammar collections simulate the skipping of the assigned transitions; that is, forward the stored n-tuple of strings to the next grammar collection in the order. Repeating this procedure, at the i_sth grammar collection we obtain a configuration which corresponds to the configuration in Γ obtained after applying the above sequence of transitions. Then, through a communication chain formed by components G_k^M, $1 \leqslant k \leqslant i_s$, word w can be transferred to the master component of Γ'. Since the grammar collections work in parallel and after finishing the simulation of the execution or the skipping of the corresponding transition, they forward the resulted strings to the next grammar collection in the order, the n-tuples of strings which "travel" through the grammar collections correspond to configurations of Γ, no other n-tuple of strings can be obtained. The assistant components of the master grammar, G_k^M, $2 \leqslant k \leqslant t$ form another chain through communication, but no terminal word can be selected by any of them which does not belong to the language generated by Γ. Thus, Γ' and Γ generate the same language. Moreover, each component of Γ' has at most 7 productions. □

The reader can observe that the number of non-terminals of the components depends on the number of non-terminals in the simulated productions. Thus, if the length of the productions is limited by a constant, a bound for the number of non-terminals appearing in the components can be given.

Theorem 3.2. *For each context-free returning PC grammar system Γ we can construct an equivalent context-free returning PC grammar system Γ' such that $CVar(\Gamma') \leqslant 8$ and $CSymb(\Gamma') \leqslant 22$.*

Proof. We modify the construction of the proof of the previous theorem in such a way, that each rule has at most two symbols on its right-hand side, then the statement of the theorem holds.

Let for each $\bar{r}_k$, $1 \leqslant k \leqslant t$, the length of the longest right-hand side of the rules of $\bar{r}_k$ be denoted by $mrlg(\bar{r}_k)$. Now for each k, $1 \leqslant k \leqslant t$, we modify the construction of the executing components of each grammar collection described in the proof of the previous theorem. The number of the executing components will be $2 \cdot n \cdot mrlg(\bar{r}_k)$, the execution of one rule will be simulated by several components, each of them introducing only one symbol of the given right-hand side together with a query symbol to collect the rest of the right-hand side in a "communication chain". The idea is similar to the proof of Theorem 4.3 of [4].

For each k, $1 \leqslant k \leqslant t$, we have $P^{\text{exe}}_{k,i,j}$ and $P^{a}_{k,i,j}$ where $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant mrlg(\bar{r}_k)$, where for each i, $1 \leqslant i \leqslant n$,

$$
\begin{aligned}
P^{\text{exe}}_{k,i,1} &= \{S \to Q^{\text{sel}}_{k}, E \to E', E' \to Q^{\text{prep}}_{k,i}, F \to F', F' \to F'', F'' \to S\} \\
&\quad \cup \{[X] \to A'_{i,1} B_{i,1} \mid r_{k,i} \neq \#\}, \\
P^{\text{exe}}_{k,i,j} &= \{S \to Q^{\text{sel}}_{k}, E \to E', E' \to E'', F \to F', F' \to F'', F'' \to S\} \\
&\quad \cup \{E'' \to A'_{i,j} B_{i,j}\}
\end{aligned}
$$

for $2 \leqslant j \leqslant mrlg(\bar{r}_k)$, where if

$$r_{k,i} = X \to A_{i,1} A_{i,2} \ldots A_{i,s}, 2 \leqslant s \leqslant mrlg(\bar{r}_k),$$

then $A'_{i,j} = A_{i,j}$ for $A_{i,j} \in N \cup T$, and $A'_{i,j} = Q^{a}_{k,i,j}$ for $A_{i,j} \in K$; and $B_{i,j} = Q^{\text{exe}}_{k,i,j+1}$ for $1 \leqslant j \leqslant s-1$, and $B_{i,j} = \lambda$ for $j = s$. For $s < j \leqslant mrlg(\bar{r}_k)$, $A'_{i,j} = S$ and $B_{i,j} = \lambda$. If

$$r_{k,i} = X \to A_i,$$

then $B_{i,j} = \lambda$ for $1 \leqslant j \leqslant mrlg(\bar{r}_k)$, and $A'_{i,1} = A_i$ for $A_i \in N \cup T$, or $A'_{i,1} = Q^{a}_{k,i,1}$ for $A_i \in K$; and for $2 \leqslant j \leqslant mrlg(\bar{r}_k)$, $A'_{i,j} = S$. If

$$r_{k,i} = X \to \lambda,$$

then $A'_{i,1} = \lambda$; and $A'_{i,j} = S$ for $2 \leqslant j \leqslant mrlg(\bar{r}_k)$, $B_{i,j} = \lambda$ for $1 \leqslant j \leqslant mrlg(\bar{r}_k)$. If

$$r_{k,i} = \#,$$

then $A'_{i,j} = S$, $B_{i,j} = \lambda$ for $2 \leqslant j \leqslant mrlg(\bar{r}_k)$.

We also have

$$P^a_{k,i,j} = \{S \to Q^{\mathrm{sel}}_k, E \to E', E' \to E'', F \to F', F' \to F'', F'' \to S\} \cup \{E'' \to X_{i,j}\}$$

for $1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant mrlg(\bar{r}_k)$, where $X_{i,j} = Q^{\mathrm{exe}}_{k,l,1}$ if $r_{k,i} = X \to A_1 A_2 \ldots A_s$ and $A_j = Q_l$, otherwise $X_{i,j} = S$.

To see how this modified system works, consider the following. For each transition $\bar{r}_k$, the effect of applying rules $r_{k,i},\ 1 \leqslant i \leqslant n$, is reproduced with $2 \cdot mrlg(\bar{r}_k)$ number of components, each rule in $P^{\mathrm{exe}}_{k,i,j},\ 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant mrlg(\bar{r}_k)$, introducing the jth symbol of the right-hand side of $r_{k,i}$ and a query to collect the rest of the right-hand side in a "communication chain". If the right-hand side of $r_{k,i}$ has a query symbol Q_l as the jth symbol, then instead of Q_l, the symbol $Q^a_{k,i,j}$ is introduced, querying component $G^a_{k,i,j}$, where $Q_{k,l,1}$ is present. This query symbol $Q_{k,l,1}$ will be replaced by the already collected sentential form which can be passed on to replace $Q^a_{k,i,j}$.

Thus, after the communication sequence following the application of the rules of $P^{\mathrm{exe}}_{k,i,j}$ and $P^a_{k,i,j},\ 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant mrlg(\bar{r}_k)$, the system have reproduced the effect of $\bar{r}_k$ with rules having at most two symbols on the right-hand side; that is, by using at most 8 non-terminals and 22 symbols at each component. □

Finally, we add a remark on the behaviour of measures *CSymb*, *CProd*, and *CVar* on the class of recursively enumerable languages. We say that a size complexity measure M is bounded on a class $\mathscr{L}$ of languages with respect to a class of grammars $\mathscr{G}$, if there is a constant k such that for any language L in $\mathscr{L}$ there is a grammar G in $\mathscr{G}$ such that L can be generated by G and $M(G) \leqslant k$ holds. Since the class of context-free returning PC grammar systems generates the class of recursively enumerable languages [3], we can conclude that complexity measure *CSymb*, and thus *CProd* and *CVar*, are bounded on the class of recursively enumerable languages with respect to the class of context-free returning PC grammar systems.

References

[1] E. Csuhaj-Varjú, On size complexity of context-free returning parallel communicating grammar systems, in: C. Martin-Vide, V. Mitrana (Eds.), Where Mathematics, Computer Science, Linguistics and Biology Meet, Kluwer Academic Publishers, Dordrecht, 2001, pp. 37–49.

[2] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun, Grammar Systems: A Grammatical Approach to Distribution and Cooperation, Gordon and Breach Science Publisher, Yverdon, 1994.

[3] E. Csuhaj-Varjú, Gy. Vaszil, On the computational completeness of contex-free parallel communicating grammar systems, Theoret. Comput. Sci. 215 1–2 (1999) 349–358.

[4] E. Csuhaj-Varjú, Gy. Vaszil, On context-free parallel communicating grammar systems: synchronization, communication, and normal forms. Theoret. Comput. Sci. (2001) 511–538.

[5] J. Dassow, Gh. Păun, G. Rozenberg, in: G. Rozenberg, A. Salomaa (Eds.), Grammar Systems, Handbook of Formal Languages, Vol. 2, Springer, Berlin-Heidelberg-New York, 1997, pp. 155–213 (Chapter 4).

[6] S. Dumitrescu, Gh. Păun, On the power of parallel communicating grammar systems with right-linear components, Inform. Théorique et Appl. R.A.I.R.O 31 (4) (1997) 331–354.

[7] Gh. Păun, Parallel communicating grammar systems: recent results, open problems, Acta Cybernet. 12 (1996) 381–395.
[8] Gh. Păun, L. Santean, Parallel communicating grammar systems: the regular case, Ann. Univ. Bucharest, Ser. Matem.-Inform. 38 (2) (1989) 55–63.
[9] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. I–II–III. Springer, Berlin-Heidelberg-New York, 1997.
[10] A. Salomaa, Formal Languages, Academic Press, New York, 1973.
[11] Gy. Vaszil, On parallel communicating Lindenmayer systems, in: Gh. Păun, A. Salomaa (Eds.), Grammatical Models of Multi-Agent Systems, Gordon and Breach Science Publishers, Amsterdam, 1999, pp. 99–112.