ELSEVIER

Theoretical Computer Science 209 (1998) 319–329

Theoretical Computer Science

On simulating non-returning PC grammar systems with returning systems

György Vaszil*

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

Received December 1996; received in revised form March 1997
Communicated by A. Salomaa

Abstract

Non-returning PC grammar systems with n context free components are simulated in [3] by returning PC grammar systems with $4n^2 - 3n + 1$ components. In this paper we reduce the number of components of the simulating system by using a different simulating construction. The number of simulating components can be further decreased if the queries appearing in some of the components satisfy a simple condition, which all queries in regular and linear grammars naturally do. This number can be as low as $2n$, while in general it is still $(n+1)^2$. © 1998—Elsevier Science B.V. All rights reserved

1. Introduction

Parallel communicating grammar systems (PC grammar systems) have been introduced in [7] as grammatical representations for the classroom model of distributed problem solving, which is a modification of the blackboard architecture, consisting of several agents with their own notebooks and a classroom leader (the master) operating on the blackboard, where the given problem is being solved. During this problem solving process, the agents may communicate with each other and the master, thus contributing to the overall solution.

A parallel communicating grammar system consists of several grammars working on their own sentential forms in parallel. Communication is done with the help of special symbols, called query symbols. If a query symbol Q_i appears in the sentential form of a component G_j, then this occurrence of Q_i must be replaced by the current sentential form of the component G_i. The language generated by the system consists of the terminal words appearing as sentential forms of the first component, the master.

* E-mail: vaszil@luna.aszi.sztaki.hu.

PII S0304-3975(97)00120-5

Parallel communicating grammar systems have been studied in detail over the past few years. Results can be found about their generative power in [6, 9], their size parameters in [4, 8], their computational complexity in [1]. A summary of results can be found in the monograph [2].

There are basically two modes of communication in PC grammar systems. In the so called *returning* mode, the components which have sent their sentential forms to other components begin to generate a new string (return to their axioms), while in the *non-returning* mode, they keep a copy of their string for themselves and continue working on it after the communication.

A PC grammar system is called *centralized*, if only the master grammar is allowed to introduce query symbols.

The relationship between the class of languages generated by returning and non-returning PC grammar systems have long been an important open problem in the field of PC grammars. In [5] Mihalache showed that non-returning centralized PC grammar systems can be simulated by returning but non-centralized system and finally in [3] Dumitrescu presented a simulation also for the general non-centralized case. This simulation uses $4n^2 - 3n + 1$ components, where n is the size of the simulated system, and one rewriting step of the simulated system corresponds to about $2n$ simulating rewriting steps.

In the following, we present a simulation of non-returning context-free PC grammar systems with returning systems using a method different from [3]. Our construction requires $(n+1)^2$ simulating components in general, but this can be reduced to $2n$ in the best case, if the components of the simulated system satisfy certain properties which for example regular and linear grammars naturally do. Furthermore, one rewriting step of the simulated system corresponds to only 2 simulating rewriting steps.

2. Preliminaries and definitions

The reader is assumed to be familiar with the basics of formal language theory; further details can be found in [10].

Now we recall the notion of parallel communicating grammar systems from [7], for more material see the monograph [2].

Definition 2.1. A *parallel communicating grammar system* with n components, where $n \geqslant 1$ (a PC grammar system, for short), 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, a set of rewriting rules P_i and an axiom or (a start symbol) S_i. G_1 is said to be the *master* (grammar) of Γ.

Definition 2.2. Let $\Gamma = (N, K, T, G_1, \ldots, G_n)$, $n \geqslant 1$, be a PC grammar system as above.

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 performing direct derivation steps.

Definition 2.3. Let $\Gamma = (N, K, T, G_1, \ldots, G_n)$, $n \geqslant 1$, be a PC 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 next two cases holds:

1. There is no x_i which contains any query symbol, that is, $x_i \in (N \cup T)^*$ for $1 \leqslant i \leqslant n$. In this case $x_i \Rightarrow_{G_i} y_i$. (For $x_i \in T^*$ we have $x_i = y_i$.)

2. There is some x_i, $1 \leqslant i \leqslant n$, which contains at least one occurrence of query symbols. Let x_i be of the form $x_i = z_1 Q_{i_1} z_2 Q_{i_2}, \ldots, z_t Q_{i_t} z_{t+1}$, where $z_j \in (N \cup T)^*, 1 \leqslant j \leqslant t+1$ and $Q_{i_l} \in K$, $1 \leqslant l \leqslant t$. In this case $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$ [and $y_{i_l} = S_{i_l}$, $1 \leqslant l \leqslant t$], where x_{i_l}, $1 \leqslant l \leqslant t$, does not contain any query symbol. If some x_{i_l} contains at least one occurrence of query symbols, then $y_i = x_i$. For all i, $1 \leqslant i \leqslant n$, for which y_i is not specified above, $y_i = x_i$.

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

The second case describes a communication: if some query symbol, say Q_i, appears in a sentential form, then rewriting stops and a communication step must be performed. This means that Q_i must be replaced by the current sentential form of component G_i, say x_i, supposing that x_i does not contain any query symbol. (Only strings without query symbols can be communicated.) If this sentential form also contains some query symbols, then first these symbols must be replaced with the requested sentential forms. If this condition cannot be fulfilled (a circular query appeared), then the derivation is blocked. For example, the configuration $(x_1,\ldots,x_{i_1} Q_j x_{i_2},\ldots,x_{j_1} Q_i x_{j_2},\ldots,x_n)$ blocks the derivation by introducing a circular query. Let $\Rightarrow_{rew}$ and $\Rightarrow_{com}$ denote a rewriting and a communication step respectively.

If the sentential form of a component was communicated to another, this component can continue its own work in two ways: in so-called *returning* systems, the component must return to its axiom and begin to generate a new string (as indicated by the words in square brackets under case 2). In *non-returning* systems the components do not return to their axiom, but continue to process the current string.

A system is *centralized* if only the component G_1 is allowed to introduce query symbols, otherwise it is *non-centralized.*

The phrase *communication step* is used to denote the process of satisfying the query symbols, which can be satisfied in "parallel". For example the communication prescribed by (Q_2, Q_3, α, Q_3) takes two communication steps to realise: first we

get $(Q_2, \alpha, S_3, \alpha)$, and then $(\alpha, S_2, S_3, \alpha)$. The two consecutive steps together will be referred to as a *communication sequence*.

Let $\Rightarrow^+$ and $\Rightarrow^*$ denote the transitive, and the reflexive, transitive closure of $\Rightarrow$.

Definition 2.4. The *language* generated by a PC grammar system Γ is

$$L(\Gamma) = \{w \in T^* | (S_1, \ldots, S_n) \Rightarrow^* (w, \alpha_2, \ldots, \alpha_n)\}.$$

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

Let PC_nX and NPC_nX denote the classes of returning and non-returning PC grammar systems with at most n components of type X, respectively, where $X \in \{RL, LIN, CF\}$ and $n \geqslant 1$. The classes of languages generated by such systems are denoted by $\mathscr{L}(PC_nX)$ and $\mathscr{L}(NPC_nX)$, respectively.

Now before we proceed, let us take a closer look at the difference between the returning and non-returning mode of communication.

3. The two modes of communication

After a communication is performed, not only the components which send their current strings, but also those components which receive these strings may finish the communication with different sentential forms in the returning or non-returning modes. This is due to the fact, that all query symbols occurring in one string must be rewritten in the same communication step. This requirement makes it possible, that a query symbol Q_i is replaced by some string α in the non-returning mode while in the returning mode it is replaced by S_i, since α may no longer be available when the replacement of Q_i becomes possible.

For example the query (Q_2Q_3, Q_3, a) is satisfied in the non-returning mode with the following two steps:

$(Q_2Q_3, Q_3, a) \Rightarrow_{com} (Q_2Q_3, a, a) \Rightarrow_{com} (aa, a, a)$, while in the returning mode it is satisfied with

$(Q_2Q_3, Q_3, a) \Rightarrow_{com} (Q_2Q_3, a, S_3) \Rightarrow_{com} (aS_3, S_2, S_3)$, producing a different sentential form in the first component.

There are special cases, when the mode of communication does not make any difference in the resulting sentential forms of those components which only receive strings during a communication (Components like G_1 in the example above.) One of them is the following:

If occurences of only one query symbol can be present in one sentential form at the same time, then all the occurrences of a certain query symbol appearing in any sentential form are replaced in the same communication step. It does not matter whether the component which sends its string returns to the axiom or not, since after the sentential form has been sent, there are no other communication symbols present requesting this

same string. In other words, all components send their sentential forms at most once during a communication sequence.

In the following we will simplify the simulation of non-returning PC grammar systems according to this observation. In our construction the number of simulating component grammars is based upon not only the size of the simulated system, but also the "complexity" of the communications the components are capable of.

In general we will need $(n+1)^2$ simulating components, but it can be less if the simulated system has at least one component, which is communicating only in the above described homogeneous way. First we present the formal definition of this homogeneousity.

Definition 3.1. Consider a PC grammar system $(N, K, T, G_1, \ldots, G_n)$ with n components $G_1, \ldots, G_n$, non-terminal and terminal alphabets N and T, query symbols $K = \{Q_1, \ldots, Q_n\}$.

By the word *query* we refer to a sentential form containing at least one query symbol. A query is satisfied through *communication* replacing the query symbols with the requested sentential forms. This may be done in one or more *communication steps.*

Let us call a query *homogeneous*, if all query symbols contained in the sentential form request the same string, which means that it is of the form $\alpha_1 Q_i \alpha_2 Q_i \ldots \alpha_{t-1} Q_i \alpha_t$, where $1 \leqslant i \leqslant n$ and $2 \leqslant t$.

Otherwise a query is *non-homogeneous*, then it is of the form $\alpha_1 Q_{i_1} \alpha_2 Q_{i_2} \ldots \alpha_{t-1} Q_{i_{t-1}} \alpha_t$ where for all $1 \leqslant j \leqslant t$, $1 \leqslant i_j \leqslant n$, $3 \leqslant t$ and at least two query symbols are different, there exists $1 \leqslant j, k \leqslant t$ for which $Q_{i_j} \neq Q_{i_k}$.

A *component with non-homogeneous queries* is a component grammar G_i, $1 \leqslant i \leqslant n$, which is capable of introducing non-homogeneous queries, it has at least one rule of the form $X \to \alpha Q_i \beta Q_j \delta$, where $i \neq j$.

4. The simulation

Now we present the basic idea of our simulating construction in the following two lemmas. We show how a non-returning communication sequence on n components can be simulated by a returning communication sequence with $n(n+2)$ components.

Lemma 4.1. *Consider the PC grammar system* $\Gamma = (N, K, T, G_1, \ldots, G_n)$ *and the non-returning communication sequence in* Γ:

$$(\alpha_1, \ldots, \alpha_n) \Rightarrow_{com}^{+} (\beta_1, \ldots, \beta_n), \text{ where } \alpha_i, \beta_i \in (N \cup T \cup K)^*, \ 1 \leqslant i \leqslant n.$$

Let $\Gamma' = (N', K, T, G'_1, \ldots, G'_n, G'_{n+1}, \ldots, G'_{2n}, G'_{1,1}, \ldots, G'_{1,n}, \ldots, G'_{n,1}, \ldots, G'_{n,n})$ *and let*

$$(Q_{n+1}, \ldots, Q_{2n}, \alpha'_1, \ldots, \alpha'_n, Q_{n+1}, \ldots, Q_{2n}, \ldots, Q_{n+1}, \ldots, Q_{2n}) \Rightarrow_{com}^{+}$$

$$(\gamma_1, \ldots, \gamma_n, S_{n+1}, \ldots, S_{2n}, \delta_{1,1}, \ldots, \delta_{1,n}, \ldots, \delta_{n,1}, \ldots, \delta_{n,n})$$

be a returning communication sequence in Γ', *where* $\alpha'_i = \alpha_i$ *if* $\alpha_i \in (N \cup T)^*$, *or* $\alpha'_i = \alpha_{i,1} Q_{i,j_1} \alpha_{i,2} Q_{i,j_2} \dots \alpha_{i,t} Q_{i,j_t} \alpha_{i,t+1}$ *if* $\alpha_i = \alpha_{i,1} Q_{j_1} \alpha_{i,2} Q_{j_2} \dots \alpha_{i,t} Q_{j_t} \alpha_{i,t+1}$, $\alpha_{i,k} \in (N \cup T)^*$, $1 \leqslant k \leqslant t+1$, $1 \leqslant j_l \leqslant n$, $1 \leqslant l \leqslant t$. *Then* $\gamma_i = \beta_i$, $1 \leqslant i \leqslant n$.

Proof. If $|\alpha_i|_K = 0$ then $\beta_i = \alpha_i$. In this case $\alpha'_i = \alpha_i$, $\gamma_i = \alpha'_i$, so $\gamma_i = \beta_i$. If $\alpha_i = \alpha_{i,1} Q_{j_1} \alpha_{i,2} Q_{j_2} \dots \alpha_{i,t} Q_{j_t} \alpha_{i,t+1}$, $\alpha_{i,k} \in (N \cup T)^*$, $1 \leqslant k \leqslant t+1$ then $\beta_i = \alpha_{i,1} \beta_{j_1} \alpha_{i,2} \beta_{j_2} \dots \alpha_{i,t} \beta_{j_t} \alpha_{i,t+1}$. In this case $\alpha'_i = \alpha_{i,1} Q_{i,j_1} \alpha_{i,2} Q_{i,j_2} \dots \alpha_{i,t} Q_{i,j_t} \alpha_{i,t+1}$. All queries of α_i are redirected in α'_i to the ith n-tuple of assistant grammars $G'_{i,1}, \dots, G'_{i,n}$. When the sentential form of a component G'_{n+i} is available for communication which means it contains no query symbols, it is sent to G'_i and n copies of it appear as sentential forms of all $G'_{j,i}$, $1 \leqslant j \leqslant n$. These copies are available for further requests by $G'_{n+1}, \dots, G'_{2n}$. From these considerations it follows that $\gamma_i = \alpha_{i,1} \beta_{j_1} \alpha_{i,2} \beta_{j_2} \dots \alpha_{i,t} \beta_{j_t} \alpha_{i,t+1} = \beta_i$. □

Now we show how to decrease the number of components in the simulating returning communication sequence.

Lemma 4.2. *Consider the PC grammar system* $\Gamma = (N, K, T, G_1, \dots, G_n)$ *and the non-returning communication sequence in* Γ:

$$(\alpha_1, \dots, \alpha_n) \Rightarrow^+_{com} (\beta_1, \dots, \beta_n), \text{ where } \alpha_i, \beta_i \in (N \cup T \cup K)^*, \ 1 \leqslant i \leqslant n,$$

and let $i_1, i_2, \dots, i_k$, $k \leqslant n$ *denote the indices of those sentential forms which introduce non-homogeneous queries.*

Let $\Gamma' = (N', K, T, G'_1, \dots, G'_n, G'_{n+1}, \dots, G'_{2n}, G'_{i_1,1}, \dots, G'_{i_1,n}, \dots, G'_{i_k,1}, \dots, G'_{i_k,n})$ *and let*

$$(Q_{n+1}, \dots, Q_{2n}, \alpha'_1, \dots, \alpha'_n, Q_{n+1}, \dots, Q_{2n}, \dots, Q_{n+1}, \dots, Q_{2n}) \Rightarrow^+_{com}$$

$$(\gamma_1, \dots, \gamma_n, S_{n+1}, \dots, S_{2n}, \delta_{i_1,1}, \dots, \delta_{i_1,n}, \dots, \delta_{i_k,1}, \dots, \delta_{i_k,n})$$

be a returning communication sequence in Γ', *where* $\alpha'_i = \alpha_i$ *if* $\alpha_i \in (N \cup T)^*$, *or* $\alpha'_i = \alpha_{i,1} Q_{n+j} \alpha_{i,2} Q_{n+j} \dots \alpha_{i,t} Q_{n+j} \alpha_{i,t+1}$ *if* $\alpha_i = \alpha_{i,1} Q_j \alpha_{i,2} Q_j \dots \alpha_{i,t} Q_j \alpha_{i,t+1} (i \notin \{i_1, i_2, \dots, i_k\})$ *and* $\alpha'_{i_j} = \alpha_{i_j,1} Q_{i_j,l_1} \alpha_{i_j,2} Q_{i_j,l_2} \dots \alpha_{i_j,t} Q_{i_j,l_t} \alpha_{i_j,t+1}$ *if* $\alpha_{i_j} = \alpha_{i_j,1} Q_{l_1} \alpha_{i_j,2} Q_{l_2} \dots \alpha_{i_j,t} Q_{l_t} \alpha_{i_j,t+1}$, $\alpha_{i_j,m} \in (N \cup T)^*$, $1 \leqslant m \leqslant t+1$, $1 \leqslant l_r \leqslant n$, $1 \leqslant r \leqslant t$ *for all* $i_j \in \{i_1, i_2, \dots, i_k\}$, $j \leqslant k$. *Then* $\gamma_i = \beta_i$, $1 \leqslant i \leqslant n$.

Proof. If $i \notin \{i_1, i_2, \dots, i_k\}$ then there are two possible cases: Either α_i does not contain a query at all or $\alpha_i = \alpha_{i,1} Q_j \alpha_{i,2} Q_j \dots \alpha_{i,t} Q_j \alpha_{i,t+1}, \alpha_{i,m} \in (N \cup T)^*$, $1 \leqslant m \leqslant t+1$, the query of α_i is homogeneous.

The statement now follows from Lemma 4.1 and from the fact that in a homogeneous query all query symbols occurring in one sentential form are replaced in one same step. □

Now we show how to construct a returning system which simulates non-returning communications in the way described above.

According to the following theorem, non-returning PC grammar systems with n context-free components can be simulated by returning systems with $(n+1)^2$

components but this number can be reduced if at least one of the simulated components is not capable of introducing non-homogeneous queries. If none of them introduces non-homogeneous queries, then $2n$ simulating components are enough.

Theorem 4.3. *If $L \in \mathscr{L}(NPC_nCF)$ and $\Gamma \in NPC_nCF$ generating L, then $L = L(\Gamma')$, where $\Gamma' \in PC_{n(k+2)+1}CF$ and $k \leqslant n$ is the number of components with non-homogeneous queries in Γ.*

Proof. We construct a returning context-free PC grammar system Γ' which simulates the behaviour of Γ. The idea of the simulation is the following. Γ' contains the components: $G'_1 \dots G'_n$ and $G'_{n+1} \dots G'_{2n}$. The rules $X \to \alpha$ of each G_i, $1 \leqslant i \leqslant n$, of Γ are broken into two parts $X \to [X]$ and $[X] \to \alpha$. The first part is contained by the components $G'_1, \dots, G'_n$, the second is by $G'_{n+1}, \dots, G'_{2n}$.

During the simulation each of $G'_1, \dots, G'_n$ select a rule to be used, by applying its first part, while the components $G'_{n+1}, \dots, G'_{2n}$ introduce query symbols, so they can receive these "partly" rewritten strings after a communication step. Now the application of the chosen rules is finished in $G'_{n+1}, \dots, G'_{2n}$ by applying their second part, while $G'_1, \dots, G'_n$ get ready to receive the strings again by introducing the appropriate query symbols.

If the second parts of the applied rules do not introduce queries in $G'_{n+1}, \dots, G'_{2n}$, then the strings are communicated to $G'_1, \dots, G'_n$ and the process can continue in the same manner by chosing a next rule. If this is not the case and the sentential forms of $G'_{n+1}, \dots, G'_{2n}$ also contain queries which must be satisfied, then the rest of the components (the assistant grammars) aid the simulation of the non-returning communication sequence by introducing further queries based on the idea introduced in Lemmas 4.1 and 4.2.

The last component G'_a is used to force a restart of the assistant grammars by querying them after the simulation of the non-returning communication sequence is finished.

Now let us see how the simulating PC grammar system is constructed. Let $\Gamma = (N, K, T, G_1, \dots, G_n) \in NPC_nCF$ with non-terminal alphabet N, set of query symbols K, terminal alphabet T and n context-free components $G_1, \dots, G_n$. Let $i_1, i_2, \dots, i_k$ denote the indices of those components of Γ which have non-homogeneous queries, and let us assume that none of the rules of Γ contains start symbols of the components on the right side. This assumption does not mean any loss of generality, since Γ is non-returning, so we can add the rule $S'_i \to S_i$ to all components and make S'_i the new start symbol if necessary. Let the simulating system be the following: $\Gamma' = (N', K', T, G'_1, \dots, G'_n, G'_{n+1}, \dots, G'_{2n}, G'_{i_1,1}, \dots, G_{i_1,n}, \dots, G'_{i_k,1}, \dots, G'_{i_k,n}, G'_a)$ where the set of non-terminals and the set of rules are

$$N' = \{X, [X] \mid X \in N\} \cup \{S_\alpha \mid G'_\alpha \textit{ is a component of } \Gamma'\}$$
$$\cup \{S', S'', S'_{i_j,l} \mid 1 \leqslant j \leqslant k, 1 \leqslant l \leqslant n\},$$

$$P_i' = \{S_i \to Q_{n+i}\} \cup \{X \to [X] \mid X \to \alpha \in P_i\}$$

$$\textit{for } 1 \leqslant i \leqslant n,$$

$$P_i' = \{S_i \to Q_{i-n}\} \cup \{[X] \to \alpha \mid X \to \alpha \in P_{i-n}, \alpha \in (N \cup T)^*\}$$

$$\cup \{[X] \to \alpha_1 Q_{j+n} \alpha_2 \ldots Q_{j+n} \alpha_t \mid X \to \alpha_1 Q_j \alpha_2 \ldots Q_j \alpha_{t+1} \in P_{i-n},$$

$$\alpha_l \in (N \cup T)^*,\ 1 \leqslant l \leqslant t+1\}$$

$$\cup \{[X] \to \alpha_1 Q_{i,j_1} \alpha_2 \ldots Q_{i,j_t} \alpha_{t+1} \mid X \to \alpha_1 Q_{j_1} \alpha_2 \ldots Q_{j_t} \alpha_{t+1} \in P_{i-n},$$

$$\alpha_l \in (N \cup T)^*,\ 1 \leqslant l \leqslant t+1,\ j_l \neq j_m \textit{ for some } l, m\}$$

$$\textit{for } n+1 \leqslant i \leqslant 2n,$$

$$P_{i_j,l}' = \{S_{i_j,l} \to S_{i_j,l}', S_{i_j,l}' \to Q_{l+n}, S_{i_j,l} \to Q_{l+n}\}$$

$$\cup \{X \to [X] \mid X \in N\}$$

$$\textit{for } 1 \leqslant j \leqslant k,\ 1 \leqslant l \leqslant n$$

$$P_a' = \{S_a \to S', S' \to S'', S'' \to Q_{i_1,1} Q_{i_1,2} \ldots Q_{i_k,n} S'\}.$$

Now we show how a rewriting step and the following communication sequence of Γ is simulated by Γ'.

For any string α, let $[\alpha]$ denote the same string with one of its non-terminals in square brackets, that is $[\alpha] = \alpha_1[X]\alpha_2$ if $\alpha = \alpha_1 X \alpha_2, \alpha_1, \alpha_2 \in (N \cup T)^*$, $X \in N$. If α does not contain any non-terminals, then $[\alpha] = \alpha$.

Let us look at the simulation of the initial step first.

1.1. If Γ cannot take a single rewriting step because some component does not have any rules with the start symbol on the left side, then the corresponding components of Γ' only have rules of the form $S_i \to Q_{n+i}$, $1 \leqslant i \leqslant n$ with the start symbol on the left. These rules produce a circular query after the first rewriting step is performed, because the components $G_{n+1}', \ldots, G_{2n}'$ all use their rules $S_{n+i} \to Q_i, 1 \leqslant i \leqslant n$. Since the simulating derivation must be blocked by a circular query without generating any terminal strings, the work of Γ is simulated correctly.

1.2. If the initial step $(S_1, \ldots, S_n) \Rightarrow_{rew} (\alpha_1, \ldots, \alpha_n)$ is possible in Γ, then in Γ' we have the following:

$$(S_1, \ldots, S_n, S_{n+1}, \ldots, S_{2n}, S_{i_1,1}, \ldots, S_{i_k,n}, S_a) \Rightarrow_{rew}$$

$$(\delta_1, \ldots, \delta_n, Q_1, \ldots, Q_n, \delta_{i_1,1}, \ldots, \delta_{i_k,n}, S')$$

where δ_i is either $[S_i]$ or Q_{i+n} and $\delta_{i_j,l}$ is either $S_{i_j,l}'$ or Q_{l+n} $1 \leqslant i, l \leqslant n$, $1 \leqslant j \leqslant k$.

If δ_i is Q_{i+n} then the system is blocked by a circular query and if $\delta_{i_j,l}$ is Q_{l+n} then no more rewriting is possible, the system is blocked again. Therefore we must have

$$([S_1],\dots,[S_n],Q_1,\dots,Q_n,S'_{i_1,1},\dots,S'_{i_k,n},S') \Rightarrow_{com}$$

$$(S_1,\dots,S_n,[S_1],\dots,[S_n],S'_{i_1,1},\dots,S'_{i_k,n},S') \Rightarrow_{rew}$$

$$(\delta_1,\dots,\delta_n,\alpha'_1,\dots,\alpha'_n,Q_{n+1},\dots,Q_{2n},\dots,Q_{n+1},\dots,Q_{2n},S''),$$

where δ_i, $1 \leqslant i \leqslant n$ is the same as above, α'_i and $\alpha_i, 1 \leqslant i \leqslant n$, differ only in the indices of the query symbols they contain. (If α_i is a homogeneous query containing Q_j then α'_i contains Q_{j+n}, if α_i is a non-homogeneous query containing Q_j then α'_i contains $Q_{i,j}$.) If α_i is not a query then $\alpha'_i = \alpha_i$.

If δ_i is $[S_i]$ then no more rewriting is possible, the system is blocked.

If no communication follows the first rewriting step in Γ, then the α'_i sentential forms are sent to $G'_1,\dots,G'_n$ and the simulation of the next rewriting step starts.

If the α'_i sentential forms introduce queries and $(\alpha_1,\dots,\alpha_n) \Rightarrow^+_{com} (\beta_1,\dots,\beta_n)$ holds in Γ, then by Lemmas 4.1 and 4.2 we know that the configuration of Γ' above simulates the non-returning communication sequence of Γ with returning communications producing

$$(\beta_1,\dots,\beta_n,S_{n+1},\dots,S_{2n},\delta_{i_1,1},\dots,\delta_{i_1,n},\dots,\delta_{i_k,1},\dots,\delta_{i_k,n},S'')$$

through a series of communication steps. The $\delta_{i_j,l}$ "garbage" will be removed after the next rewriting step by G'_a.

2. Now let us see, how further rewriting steps and communications are simulated. Let us assume that $(\alpha_1,\dots,\alpha_n) \Rightarrow_{rew} (\beta_1,\dots,\beta_n)$ holds in Γ. Now Γ' starts from a configuration

$$(\alpha_1,\dots,\alpha_n,S_{n+1},\dots,S_{2n},\delta_{i_1,1},\dots,\delta_{i_1,n},\dots,\delta_{i_k,1},\dots,\delta_{i_k,n},\delta S''),$$

where $\delta_{i_j,l} \in (N \cup T)^+$. After a rewriting step we get

$$([\alpha_1],\dots,[\alpha_n],Q_1,\dots,Q_n,[\delta_{i_1,1}],\dots,[\delta_{i_1,n}],\dots,[\delta_{i_k,1}],\dots,[\delta_{i_k,n}],\delta Q_{i_1,1}\dots Q_{i_k,n}S').$$

Now a communication follows and then

$$(S_1,\dots,S_n,[\alpha_1],\dots,[\alpha_n],S_{i_1,1},\dots,S_{i_1,n},\dots,S_{i_k,1},\dots,S_{i_k,n},\delta'S') \Rightarrow_{rew}$$

$$(\delta_1,\dots,\delta_n,\beta'_1,\dots,\beta'_n,\delta_{i_1,1},\dots,\delta_{i_1,n},\dots,\delta_{i_k,1},\dots,\delta_{i_k,n},\delta'S''),$$

where $\delta_i, 1 \leqslant i \leqslant n$ is either $[S_i]$ or Q_{i+n}, β'_i and $\beta_i, 1 \leqslant i \leqslant n$, differ only in the indices of the query symbols they contain and $\delta_{i_j,l}$ is either Q_{l+n} or $S'_{i_j,l}$. If δ_i is $[S_i]$ then the system is blocked.

If the β'_i sentential forms do not introduce queries, then they are communicated to $G'_1,\dots,G'_n$ and the simulation of the next rewriting step can begin, while the assistant components send their sentential forms to G'_a.

If $(\beta_1,\ldots,\beta_n)\Rightarrow^{+}_{com}(\gamma_1,\ldots,\gamma_n)$ holds in Γ then it is simulated as we know from Lemmas 4.1 and 4.2, if all $\delta_{i_j,l}$ is Q_{l+n}. Then we have

$$(Q_{n+1},\ldots,Q_{2n},\beta'_1,\ldots,\beta'_n,Q_{n+1},\ldots,Q_{2n},\ldots,Q_{n+1},\ldots,Q_{2n},\delta'S'')$$

and then

$$(\gamma_1,\ldots,\gamma_n,S_{n+1},\ldots,S_{2n},\delta_{i_1,1},\ldots,\delta_{i_1,n},\ldots,\delta_{i_k,1},\ldots,\delta_{i_k,n},\delta'S'').$$

Now the simulation of the next rewriting step can start.

If any of the $\delta_{i_j,l}$ sentential forms is $S'_{i_j,l}$ and $S'_{i_j,l}$ becomes part of a sentential form $\gamma_j, 1\leqslant j\leqslant n$, during the communication sequence, then this $S_{i_j,l}$ non-terminal is never going to be rewritten, so it is impossible to derive words not in $L(\Gamma)$ this way.

This completes our proof. □

Example. Let us look at the example used in Section 3 again:

Let $\Gamma=(\{S_1,S_2,S_3\},\{Q_1,Q_2,Q_3,\},\{a\},G_1,G_2,G_3)$ be a non-returning PC grammar system with

$$P_1=\{S_1\rightarrow Q_2Q_3\},$$

$$P_2=\{S_2\rightarrow Q_3\},$$

$$P_3=\{S_3\rightarrow a\}.$$

The only possible derivation in Γ is

$$(S_1,S_2,S_3)\Rightarrow_{rew}(Q_2Q_3,Q_3,a)\Rightarrow_{com}(Q_2Q_3,a,a)\Rightarrow_{com}(aa,a,a).$$

If we construct Γ' as described above, it simulates this derivation through the following steps:

$$(S_1,S_2,S_3,S_4,S_5,S_6,S_{1,1},S_{1,2},S_{1,3},S_a)\Rightarrow_{rew}$$

$$([S_1],[S_2],[S_3],Q_1,Q_2,Q_3,S'_{1,1},S'_{1,2},S'_{1,3},S')\Rightarrow_{com}$$

$$(S_1,S_2,S_3,[S_1],[S_2],[S_3],S'_{1,1},S'_{1,2},S'_{1,3},S')\Rightarrow_{rew}$$

$$(Q_4,Q_5,Q_6,Q_{1,2}Q_{1,3},Q_6,a,Q_4,Q_5,Q_6,S'')\Rightarrow_{com}$$

$$(Q_4,Q_5,a,Q_{1,2}Q_{1,3},a,S_6,Q_4,Q_5,a,S'')\Rightarrow_{com}$$

$$(Q_4,a,a,Q_{1,2}Q_{1,3},S_5,S_6,Q_4,a,a,S'')\Rightarrow_{com}$$

$$(Q_4,a,a,aa,S_5,S_6,Q_4,a,a,S'')\Rightarrow_{com}$$

$$(aa,a,a,S_4,S_5,S_6,aa,S_{1,2},S_{1,3},S'').$$

When the simulating communication sequence is complete, $G'_1,\ldots,G'_3$ contains the expected result, $G'_4,\ldots,G'_6$ contain their start symbols $S_4,\ldots,S_6$, and the sentential forms of the assistant grammars will be removed by G'_a after the next rewriting step.

5. Conclusions

Regular and linear PC grammar systems never introduce non-homogeneous queries since their sentential forms contain only one non-terminal, therefore our construction works in these cases without any assistant grammars which makes the number of simulating components only $2n$.

The same holds in the context-free case if the system is centralized, even if its queries are non-homogeneous. This is easy to see, if we consider that communications in a centralized system can only consist of one communication step, since every requested sentential form is always available (only the master grammar can introduce queries), there is no need to save the intermediate results of communications.

In these cases the size of the simulating system produced by our method is therefore linear, while in the general context-free case it remains quadratic compared to the size of the simulated system.

References

[1] L. Cai, The computational complexity of PCGS with regular components, in: J. Dassow, G. Rozenberg, A. Salomaa (Eds.), Proc. Developments in Language Theory II, World Scientific, Singapore, 1995, pp. 79–87.

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

[3] S. Dumitrescu, Non-returning PC grammar systems can be simulated by returning systems, Theoret. Comput. Sci. 165 (1996) 463–474.

[4] J. Kari, L. Santean, The impact of the number of cooperating grammars on the generative power, Theoret. Comput. Sci. 98 (1992) 249–263.

[5] V. Mihalache, On parallel communicating grammar systems with context-free components, in: Gh. Păun (Ed.), Mathematical Linguistics and Related Topics, The Publ. House of the Romanian Academy, Bucharest, 1995, pp. 258–270.

[6] Gh. Păun, Parallel communicating grammar systems, the context-free case, Found. Control Engrg. 14 (1) (1989) 39–50.

[7] Gh. Păun, L. Santean, Parallel communicating grammar systems, the regular case, Ann. Univ. Bucharest, Ser. Matem.-Inform. 38 (2) (1989) 55–63.

[8] Gh. Păun, On the syntactic complexity of parallel communicating grammar systems, Kybernetika 28 (2) (1992) 146–157.

[9] Gh. Păun, A. Salomaa, S. Vicolov, On the generative capacity of parallel communicating grammar systems, Intern. J. Comput. Math. 45 (1992) 49–59.

[10] A. Salomaa, Formal Languages, Academic Press, New York, 1973.