On the Computational Completeness of Context-free Parallel Communicating Grammar Systems *

Erzsébet Csuhaj-Varjú
and
György Vaszil

Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende utca 13-17, 1111 Budapest, Hungary
csuhaj/vaszil@luna.aszi.sztaki.hu

Abstract

We prove that all recursively enumerable languages can be generated by context-free returning parallel communicating grammar systems by showing how the parallel communicating grammars can simulate two-counter machines, a class of Turing machine variants which is known to be computationally complete. Moreover, we prove that systems with a bounded number of components are sufficient to reach this generative power.

1 Introduction

Parallel communicating grammar systems (PC grammar systems, in short), introduced in [8], are formal language theoretic models of parallel and distributed computation. In these systems several grammars derive their own sentential forms in parallel and their work is organized in a communicating system to generate a single language. The parallel communicating frame has the following basic properties: the work of the components is synchronized by a universal clock, each component executes one rewriting step in each time unit, and communication is done by request through special nonterminals, called query symbols, one different symbol denoting each component of the system. When a

*Research supported by the Hungarian Scientific Research Fund OTKA, Grant no. T 017105.

component introduces a query symbol in its sentential form, the rewriting process stops and one or more communication steps are performed by replacing all occuring query symbols with the current sentential forms of the corresponding component grammars supposing that those strings have no occurrence of any query symbol. If the sentential form of a component was communicated to another one, this component can continue its own work in two ways: in so-called *returning* systems, the component returns to its axiom and starts to generate a new string. In *non-returning* systems, the components do not return to their axiom, but continue processing their current sentential forms.

Parallel communicating grammar systems have been the subject of detailed study over the last years: see [7] for a summary of results and open problems.

One of the most important questions that has remained open for a long time is how large generative capacity can be obtained by returning PC grammar systems with context-free components.

In this paper we show that returning parallel communicating grammar systems with eleven context-free components can simulate two-counter machines, a restricted but computationally complete class of variants of Turing machines ([4]). Then, the class of languages generated by context-free returning parallel communicating grammar systems is equal to the class of recursively enumerable languages. Moreover, if the recursively enumerable language does not contain the empty word, then the components of the generating PC grammar system have no erasing rule.

A recent result indenpendently obtained from our one ([6]) states that the class of languages generated by the context-free non-returning parallel communicating grammar systems is equal to the class of recursively enumerable languages, and then, by [3], that the statement is valid for the returning case, too. Our proof technique significantly differs from that one, moreover, our proof demonstrates that PC grammar systems reach computational completeness with a bounded number of components.

2 Preliminaries

The reader is assumed to be familiar with the basics of formal language theory; here we only list the necessary notions. Further details can be found in [1], [2], [5], [9].

The set of all words over an alphabet Σ and the empty word are denoted by Σ^* and ε, respectively, and the family of recursively enumerable languages by $\mathcal{L}(RE)$. $|X|$ denotes the number of elements of a finite set X, while $|w|$ and $|w|_X$ denotes the length of a word w and the number of occurences of symbols from set X in w, respectively.

In the following we recall the notion of a two-counter machine, for more details see [5] and [4]. Note, that our notations slightly differ from the customary. We chose to leave the conventions in order to avoid confusion with notations we

use for PC grammar systems.

A *two-counter machine* $M = (\Sigma \cup \{Z, B\}, E, R)$ is a 3-tape Turing machine, where Σ is an *alphabet*, E is a set of *internal states* with two distinct elements $q_0, q_F \in E$ and R is a set of *transition rules*. The machine has a read-only input tape and two semi-infinite storage tapes (the counters). The alphabet of the storage tapes contains only two symbols, Z and B (blank), while the alphabet of the input tape is $\Sigma \cup \{B\}$. R consists of transition rules of the form $(q, x, c_1, c_2, q', e_1, e_2, g)$, where $x \in \Sigma \cup \{B\}$ is the symbol scanned on the input tape in state $q \in E$ and $c_1, c_2 \in \{Z, B\}$ are the symbols scanned on the storage tapes. M enters state $q' \in E$, the counters should be modified according to $e_1, e_2 \in \{-1, 0, +1\}$, and the input head is moved according to $g \in \{0, +1\}$. If $g = 0$, then the input head does not move, if $g = +1$, then it moves one cell to the right.

The symbol Z appears initially on the cells scanned by the storage tape heads and may never appear on any other cell. An integer i can be stored by moving a tape head i cells to the right of Z. A stored number can be incremented or decremented by moving the tape head right or left. The machine is capable of checking whether a stored value is *zero* or not, by looking at the symbol scanned by the storage tape heads. If a scanned symbol is Z, then the value stored in the corresponding counter is *zero*. A word $w \in \Sigma$ is accepted by the two counter machine if the input head is scanning the last non-blank symbol on the input tape and the machine is in the accepting state q_F. Two-counter machines are computationally complete, they are just as powerful as Turing-machines, [4].

Now we recall the notion of a parallel communicating grammar system from [8], for more information see [1] and [2].

Definition 2.1 A *parallel communicating grammar system* with n components, where $n \geq 1$, (a PC grammar system, in short), 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* and $K = \{Q_1, Q_2, \ldots, Q_n\}$ is an alphabet of *query symbols*. N, T and K are pairwise disjoint sets, $G_i = (N \cup K, T, P_i, S_i)$, $1 \leq i \leq n$, called a *component* of Γ, is a usual Chomsky grammar with nonterminal alphabet $N \cup K$, terminal alphabet T, a set of productions P_i and an *axiom* (or startsymbol) S_i. G_1 is said to be the *master grammar* (or master) of Γ.

Definition 2.2 Let $\Gamma = (N, K, T, G_1, \ldots, G_n)$, $n \geq 1$, be a PC grammar system. An n-tuple $(x_1, \ldots, x_n)$, where $x_i \in (N \cup T \cup K)^*$, $1 \leq i \leq n$, 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 $\Gamma = (N, K, T, G_1, \ldots, G_n)$, $n \geq 1$, 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 hold:

1. There is no x_i which contains any query symbol, that is, $x_i \in (N \cup T)^*$ for $1 \leq i \leq n$. In this case y_i is obtained from x_i by a direct derivation step in G_i, that is $x_i \Rightarrow_{G_i} y_i$; for $x_i \in T^*$ we have $x_i = y_i$.

2. There is some $x_i,\ 1 \leq i \leq n$, 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 $z_j \in (N \cup T)^*,\ 1 \leq j \leq t+1$ and $Q_{i_l} \in K,\ 1 \leq l \leq t$. In this case, if $x_{i_l},\ 1 \leq l \leq t$ does not contain any query symbol, then $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$, and $y_{i_l} = S_{i_l},\ 1 \leq l \leq t$. If some $x_{i_l},\ 1 \leq l \leq t$ contains at least one occurrence of query symbols, then $y_i = x_i$ and also $y_{i_l} = x_{i_l},\ 1 \leq l \leq t$.

For all $i,\ 1 \leq i \leq n$, 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 symbol is 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 of some component grammar 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. The symbol 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. If this sentential form also contains 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.

Let $\Rightarrow_{rew}$ and $\Rightarrow_{com}$ denote a rewriting step and a communication step, respectively.

If the sentential form of a component was communicated to another one, this component can continue its own work in two ways: in so-called *returning* systems defined above, the component returns to its axiom and starts to generate a new string. In *non-returning* systems, the components do not return to their axioms, but continue processing their current strings.

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

Definition 2.4 Let $\Gamma = (N, K, T, G_1, ..., G_n)$ be a PC grammar system with master grammar G_1 and let $(S_1, ..., S_n)$ denote the initial configuration of Γ. The *language* generated by the PC grammar system Γ 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 appearing as sentential forms of the master grammar, G_1.

Let the class of returning PC grammar systems with at most n context-free components and the class of languages generated by these systems be denoted by PC_nCF and $\mathcal{L}(PC_nCF)$, respectively. When an arbitrary number of components is considered, we use $*$ in the subscript instead of n.

3 The power of context-free PC grammar systems

In this section we show that the class of recursively enumerable languages is equal to the class of languages generated by context-free returning PC grammar systems. Moreover, systems with a bounded number of components are sufficient to reach this generative power. If the language we would like to generate does not contain the empty word, then none of the components has erasing rules.

Theorem 3.1

$$\mathcal{L}(RE) = \mathcal{L}(PC_{11}CF) = \mathcal{L}(PC_*CF).$$

Proof: We only prove the inclusion $\mathcal{L}(RE) \subseteq \mathcal{L}(PC_{11}CF)$, because by using standard techniques it can be shown that $\mathcal{L}(PC_*CF) \subseteq \mathcal{L}(RE)$, and clearly, $\mathcal{L}(PC_{11}CF) \subseteq \mathcal{L}(PC_*CF)$.

Let $L \in \Sigma^*$ be an arbitrary language, and let $M = (\Sigma \cup \{Z, B\}, E, R)$ be a two-counter machine accepting L, with tape alphabet $\Sigma \cup \{Z, B\}$, set of internal states E and set of transition rules R, with elements $(x, q, c_1, c_2, q', e_1, e_2, g) \in R$, where $x \in \Sigma \cup \{B\}$, $q, q' \in E$, $c_1, c_2 \in \{Z, B\}$, $e_1, e_2 \in \{-1, 0, +1\}$ and $g \in \{0, +1\}$. The initial and the accepting states of M are denoted by q_0 and q_F, respectively. We construct a context-free PC grammar system Γ with *eleven* components which generates L.

First, let us assume that L is ε-free. Let

$$\Gamma = (N, K, T, G_m, G_1^{c_1}, G_2^{c_1}, G_3^{c_1}, G_4^{c_1}, G_1^{c_2}, G_2^{c_2}, G_3^{c_2}, G_4^{c_2}, G_{a_1}, G_{a_2}),$$

with

$$\begin{aligned} N &= \{[x, q, c_1, c_2, e_1, e_2], [e_1]', [e_2]', [I], [I]', \langle I\rangle, \langle x, q, c_1, c_2, e_1, e_2\rangle \mid \\ &\quad x \in \Sigma,\ q \in E,\ c_1, c_2 \in \{Z, B\},\ e_1, e_2 \in \{-1, 0, +1\}\} \cup \\ &\quad \{S, S_1, S_2, S_3, S_4, S_4^{(1)}, S_4^{(2)}, S^{(1)}, S^{(2)}, S^{(3)}, S^{(4)}\} \cup \\ &\quad \{A, C\}, \\ T &= \Sigma \cup \{a\} \end{aligned}$$

and rule sets

$$\begin{aligned} P_m &= \{S \to [I], [I] \to C,\ C \to Q_{a_1}\} \cup \\ &\quad \{\langle I\rangle \to [x, q, Z, Z, e_1, e_2] \mid \langle x, q_0, Z, Z, q, e_1, e_2, 0\rangle \in R\} \cup \\ &\quad \{\langle I\rangle \to x[y, q, Z, Z, e_1, e_2] \mid \langle x, q_0, Z, Z, q, e_1, e_2, +1\rangle \in R,\ y \in \Sigma\} \cup \\ &\quad \{\langle x, q, c_1', c_2', e_1', e_2'\rangle \to [x, q', c_1, c_2, e_1, e_2] \mid \langle x, q, c_1, c_2, q', e_1, e_2, 0\rangle \in R, \\ &\quad\ c_1', c_2' \in \{Z, B\},\ e_1', e_2' \in \{-1, 0, +1\}\} \cup \\ &\quad \{\langle x, q, c_1', c_2', e_1', e_2'\rangle \to x[y, q', c_1, c_2, e_1, e_2],\ \langle x, q_F, c_1', c_2', e_1', e_2'\rangle \to x \mid \end{aligned}$$

$$\langle x, q, c_1, c_2, q', e_1, e_2, +1\rangle \in R,\ c_1', c_2' \in \{Z, B\},$$
$$e_1', e_2' \in \{-1, 0, +1\},\ y \in \Sigma\},$$

$$\begin{aligned}
P_1^{c_1} &= \{S_1 \to Q_m,\ S_1 \to Q_4^{c_1},\ C \to Q_m\} \cup \\
&\{[x, q, c_1, c_2, e_1, e_2] \to [e_1]',\ [+1]' \to AAC,\ [0]' \to AC,\ [-1]' \to C \mid \\
&\quad x \in \Sigma,\ q \in E,\ c_1, c_2 \in \{Z, B\},\ e_1, e_2 \in \{-1, 0, +1\}\} \cup \\
&\{[I] \to [I]',\ [I]' \to AC\},
\end{aligned}$$

$$\begin{aligned}
P_2^{c_1} &= \{S_2 \to Q_m,\ S_2 \to Q_4^{c_1},\ C \to Q_m,\ A \to A\} \cup \\
&\{[x, q, Z, c_2, e_1, e_2] \to [x, q, Z, c_2, e_1, e_2],\ [I] \to [I] \mid x \in \Sigma,\ q \in E, \\
&\quad c_2 \in \{Z, B\},\ e_1, e_2 \in \{-1, 0, +1\}\},
\end{aligned}$$

$$\begin{aligned}
P_3^{c_1} &= \{S_3 \to Q_m,\ S_3 \to Q_4^{c_1},\ C \to Q_m\} \cup \\
&\{[x, q, Z, c_2, e_1, e_2] \to a,\ [x, q, B, c_2, e_1, e_2] \to [x, q, B, c_2, e_1, e_2], \\
&\quad [I] \to [I] \mid x \in \Sigma,\ q \in E,\ c_2 \in \{Z, B\},\ e_1, e_2 \in \{-1, 0, +1\}\},
\end{aligned}$$

$$P_4^{c_1} = \{S_4 \to S_4^{(1)},\ S_4^{(1)} \to S_4^{(2)},\ S_4^{(2)} \to Q_1^{c_1},\ A \to a\},$$

$$\begin{aligned}
P_1^{c_2} &= \{S_1 \to Q_m,\ S_1 \to Q_4^{c_2},\ C \to Q_m\} \cup \\
&\{[x, q, c_1, c_2, e_1, e_2] \to [e_2]',\ [+1]' \to AAZ,\ [0] \to AC,\ [-1] \to C \mid \\
&\quad x \in \Sigma,\ q \in E,\ c_1, c_2 \in \{Z, B\},\ e_1, e_2 \in \{-1, 0, +1\}\} \cup \\
&\{[I] \to [I]',\ [I]' \to AC\},
\end{aligned}$$

$$\begin{aligned}
P_2^{c_2} &= \{S_2 \to Q_m,\ S_2 \to Q_4^{c_2},\ C \to Q_m,\ A \to A\} \cup \\
&\{[x, q, c_1, Z, e_1, e_2] \to [x, q, c_1, Z, e_1, e_2],\ [I] \to [I] \mid x \in \Sigma,\ q \in E, \\
&\quad c_1 \in \{Z, B\},\ e_1, e_2 \in \{-1, 0, +1\}\},
\end{aligned}$$

$$\begin{aligned}
P_3^{c_2} &= \{S_3 \to Q_m,\ S_3 \to Q_4^{c_2},\ C \to Q_m\} \cup \\
&\{[x, q, c_1, Z, e_1, e_2] \to a,\ [x, q, c_1, B, e_1, e_2] \to [x, q, c_1, B, e_1, e_2], \\
&\quad [I] \to [I] \mid x \in \Sigma,\ q \in E,\ c_1 \in \{Z, B\},\ e_1, e_2 \in \{-1, 0, +1\}\},
\end{aligned}$$

$$P_4^{c_2} = \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}, S_4^{(2)} \to Q_1^{c_2}, A \to a\},$$

$$\begin{aligned}
P_{a_1} &= \{S \to Q_m,\ [I] \to \langle I\rangle,\ [x, q, c_1, c_2, e_1, e_2] \to \langle x, q, c_1, c_2, e_1, e_2\rangle, \\
&\quad \langle x, q, c_1, c_2, e_1, e_2\rangle \to \langle x, q, c_1, c_2, e_1, e_2\rangle,\ \langle I\rangle \to \langle I\rangle, \mid\ x \in \Sigma,
\end{aligned}$$

$$q \in E,\ c_1, c_2 \in \{Z, B\},\ e_1, e_2 \in \{-1, 0, +1\}\},$$

$$\begin{aligned} P_{a_2} \quad = \quad & \{S \to S^{(3)},\ S^{(1)} \to S^{(2)},\ S^{(2)} \to S^{(3)},\ S^{(3)} \to S^{(4)}, \\ & S^{(4)} \to Q_2^{c_1} Q_3^{c_1} Q_2^{c_2} Q_3^{c_2} S^{(1)}\}. \end{aligned}$$

The work of this system is controlled by component G_m, the master, through the nonterminals $[x, q, c_1, c_2, e_1, e_2]$, where $x \in \Sigma,\ q \in E,\ c_1, c_2 \in \{Z, B\},\ e_1, e_2 \in \{-1, 0, +1\}$. The presence of this symbol in the sentential form of G_m indicates that the two-counter machine M has entered state q, it is now scanning x on the input tape and c_1, c_2 on the two storage tapes, respectively, and the heads of the storage tapes must be moved according to e_1, e_2 before the next transition. The components $G_i^{c_1}, G_i^{c_2},\ 1 \leq i \leq 4$, are simulating the change of the contents of the counters. The number of A-s in their sentential forms corresponds to the value stored in the counters of M.

The PC grammar system Γ first introduces $[I]$ in G_m, then a series of steps follow, which initialize Γ by setting the counters to *zero*. After these steps Γ is ready to simulate the first transition of M. This is done by changing $[I]$ to $u[x', q, Z, Z, e_1, e_2]$ where M has a rule $(x, q_0, Z, Z, q, e_1, e_2, g)$. Here $u = x$ if $g = +1$ and $u = \varepsilon,\ x' = x$, if $g = 0$. If the input head moves $(g = +1)$, G_m generates x and $[x', q, Z, Z, e_1, e_2]$ indicates that M is scanning a new symbol, if the input head does not move, G_m generates no terminals and $[x, q, Z, Z, e_1, e_2]$ indicates that M is still scanning the same symbol. Now $G_2^{c_1}, G_3^{c_1}$ and $G_2^{c_2}, G_3^{c_2}$ make sure, that Z is read on the both storage tapes, by checking if the number of A-s in their sentential form is *zero* or not, while $G_1^{c_1}, G_4^{c_1}$ and $G_1^{c_2}, G_4^{c_2}$ modify the stored values according to e_1, e_2.

This way Γ checks if it is legal to enter state q by looking at the counters and also updates the counters before simulating the next transition. To simulate the next transition G_m rewrites $[x, q, c_1, c_2, e_1, e_2]$ to $u[x', q'c_1', c_2', e_1', e_2'],\ u \in \{x, \varepsilon\}$, if M has a rule $(x, q, c_1', c_2', q', e_1', e_2', g)$. Here $u = x$ if $g = +1$, and $u = \varepsilon,\ x' = x$ if $g = 0$. Now Γ has to check again if c_1' and c_2' are scanned on the counter tapes and then modify them according to e_1', e_2'. If the input head moved $(g = +1)$, the symbol x is added to the sentential form of G_m.

We now describe the functioning of Γ in details to prove that each terminating derivation of Γ corresponds to an accepting computation of M, and reversely.

Γ starts with the initial configuration

$(S, S_1, S_2, S_3, S_4, S_1, S_2, S_3, S_4, S, S)$.

After the first rewriting step, Γ enters the configuration

$([I], u_1, u_2, u_3, S_4^{(1)}, u_1', u_2', u_3', S_4^{(1)}, Q_m, S^{(3)})$,

where u_1, u_2, u_3 are either Q_m or $Q_4^{c_1}$ and u_1', u_2', u_3' are either Q_m or $Q_4^{c_2}$. If any of these symbols is not Q_m, the system is blocked after the communication so we have

$([I], Q_m, Q_m, Q_m, S_4^{(1)}, Q_m, Q_m, Q_m, S_4^{(1)}, Q_m, S^{(3)}) \Rightarrow_{com}$

$(S, [I], [I], [I], S_4^{(1)}, [I], [I], [I], S_4^{(1)}, [I], S^{(3)})$.

The next steps of the system are

$(S, [I], [I], [I], S_4^{(1)}, [I], [I], [I], S_4^{(1)}, [I], S^{(3)}) \Rightarrow_{rew}$

$([I], [I]', [I], [I], S_4^{(2)}, [I]', [I], [I], S_4^{(2)}, \langle I\rangle, S^{(4)}) \Rightarrow_{rew}$

$(C, AC, [I], [I], Q_1^{c_1}, AC, [I], [I], Q_1^{c_2}, \langle I\rangle, Q_2^{c_1} Q_3^{c_1} Q_2^{c_2} Q_3^{c_2} S^{(1)}) \Rightarrow_{com}$

$(C, S_1, S_2, S_3, AC, S_1, S_2, S_3, AC, \langle I\rangle, [I][I][I][I]S^{(1)}) \Rightarrow_{rew}$

$(Q_{a_1}, u_1, u_2, u_3, aC, u_1', u_2', u_3', aC, \langle I\rangle, [I][I][I][I]S^{(2)})$,

where u_1, u_2, u_3 are either Q_m or $Q_4^{c_1}$ and u_1', u_2', u_3' are either Q_m or $Q_4^{c_2}$. If any of these sentential form is Q_m, the system is blocked after the communication, so we have

$(Q_{a_1}, Q_4^{c_1}, Q_4^{c_1}, Q_4^{c_1}, aC, Q_4^{c_2}, Q_4^{c_2}, Q_4^{c_2}, aC, \langle I\rangle, [I][I][I][I]S^{(2)}) \Rightarrow_{com}$

$(\langle I\rangle, aC, aC, aC, S_4, aC, aC, aC, S_4, S, [I][I][I][I]S^{(2)})$.

Now the PC grammar system Γ starts to simulate the steps of M. The present configuration of Γ corresponds to M being in the initial state and storing *zero* in both of the counters. Now G_m choses the next state by introducing $u[x', q, Z, Z, e_1, e_2,]$ if M has a rule $(x, q_0, Z, Z, q, e_1, e_2, g)$, if M can enter state q from q_0 by reading input x and counter symbols Z, Z. If the input head of M moves after this transition, $(g = +1)$, then $u = x$ and a new symbol x' is scanned on the input tape, if the input head does not move, $(g = 0)$, then $u = \varepsilon$, $x' = x$, the symbol x is scanned on the input tape. That is,

$(u[x', q, Z, Z, e_1, e_2],$
$aQ_m, aQ_m, aQ_m, S_4^{(1)},$
$aQ_m, aQ_m, aQ_m, S_4^{(1)},$
$Q_m, [I][I][I][I]S^{(3)}) \Rightarrow_{com}$

$(S,$
$au[x', q, Z, Z, e_1, e_2], au[x', q, Z, Z, e_1, e_2], au[x', q, Z, Z, e_1, e_2], S_4^{(1)},$
$au[x', q, Z, Z, e_1, e_2], au[x', Z, Z, e_1, e_2], au[x', q, Z, Z, e_1, e_2], S_4^{(1)},$
$u[x', q, Z, Z, e_1, e_2], [I][I][I][I]S^{(3)})$.

Now $G_1^{c_1}, G_4^{c_1}$ and $G_1^{c_2}, G_4^{c_2}$ are going to modify the number of A-s in their sentential forms (the values stored in the counters) according to e_1, e_2. $G_1^{c_1}$ and $G_1^{c_2}$ introduce AAC, AC or C if e_1 and e_2 is $+1$, 0 or -1, and $G_4^{c_1}$, $G_4^{c_2}$ erase one of the A-s. This way the system either modifies the counters or, in the case it needs to decrement *zero*, it blocks.

The components $G_2^{c_1}, G_3^{c_1}$ and $G_2^{c_2}, G_3^{c_2}$ check whether the number of A-s in their sentential forms (the values stored in the counters) correspond to Z, or

c_1, c_2 in the general case. Now we describe how Γ checks the first counter; the second one is checked in the same way.

If $c_1 = Z$, we have a string of the form $\alpha[x', q, Z, c_2, e_1, e_2]$ in $G_2^{c_1}, G_3^{c_1}$. Now the number of A-s in α should be *zero*. If it is not the case, the system blocks because in the next step $G_3^{c_1}$ rewrites $[x', q, Z, c_2, e_1, e_2]$ to a, a terminal symbol, and has no rule to rewrite A.

If $c_1 = B$, we have $\alpha[x', q, B, c_2, e_1, e_2]$, where the number of A-s in the string α should be at least *one*. If this is not the case, the system blocks, because $G_2^{c_1}$ has only the rule $A \to A$ to rewrite this sentential form.

This process we have described is the following:

$$
\begin{aligned}
&(S,\\
&\quad au[x', q, Z, Z, e_1, e_2], au[x', q, Z, Z, e_1, e_2], au[x', q, Z, Z, e_1, e_2], S_4^{(1)},\\
&\quad au[x', q, Z, Z, e_1, e_2], au[x', q, Z, Z, e_1, e_2], au[x', q, Z, Z, e_1, e_2], S_4^{(1)},\\
&\quad u[x', q, Z, Z, e_1, e_2], [I][I][I][I]S^{(3)}) \Rightarrow_{rew}\\
&([I],\\
&\quad au[e_1]', au[x', q, Z, Z, e_1, e_2], aua, S_4^{(2)},\\
&\quad au[e_2]', au[x', q, Z, Z, e_1, e_2], aua, S_4^{(2)},\\
&\quad u\langle x', q, Z, Z, e_1, e_2\rangle, [I][I][I][I]S^{(4)}) \Rightarrow_{rew}\\
&(C,\\
&\quad \alpha C, au[x', q, Z, Z, e_1, e_2], aua, Q_1^{c_1},\\
&\quad \beta C, au[x', q, Z, Z, e_1, e_2], aua, Q_1^{c_2},\\
&\quad u\langle x', q, Z, Z, e_1, e_2\rangle, \gamma Q_2^{c_1} Q_3^{c_1} Q_2^{c_2} Q_3^{c_2} S^{(1)}) \Rightarrow_{com}\\
&(C, S_1, S_2, S_3, \alpha C, S_1, S_2, S_3, \beta C, u\langle x', q, Z, Z, e_1, e_2\rangle, \gamma' S^{(1)}) \Rightarrow_{rew}\\
&(Q_{a_1}, u_1, u_2, u_3, \alpha' C, u_1, u_2, u_3, \beta' C, u\langle x', q, Z, Z, e_1, e_2\rangle, \gamma' S^{(2)}).
\end{aligned}
$$

where u_1, u_2, u_3 are either Q_m or $Q_4^{c_1}$ and u_1', u_2', u_3' are either Q_m or $Q_4^{c_2}$, β is defined in the same way as α above, and γ, γ' are the corresponding strings consisting of $[I]$. If any of these sentential form is Q_m, the system is blocked after the communication, so we have

$$
\begin{aligned}
&(Q_{a_1}, Q_4^{c_1}, Q_4^{c_1}, Q_4^{c_1}, \alpha' C, Q_4^{c_2}, Q_4^{c_2}, Q_4^{c_2}, \beta' C, u\langle x', q, Z, Z, e_1, e_2\rangle, \gamma' S^{(2)}) \Rightarrow_{com}\\
&(u\langle x', q, Z, Z, e_1, e_2\rangle, \alpha' C, \alpha' C, \alpha' C, S_4, \beta' C, \beta' C, \beta' C, S_4, S, \gamma' S^{(2)}),
\end{aligned}
$$

where α' and β' contain the same number of A-s as stored in the counters of M. If q is the accepting state, $q = q_F$, the system can stop here by using $\langle x', q_F, Z, Z, e_1, e_2\rangle \to x'$ in G_m, or it can continue by chosing a new transition. If this new transition moves the input head of M to the right, then M leaves the symbol x' behind. In this case, x' becomes part of the string generated by Γ by using $\langle x', q, Z, Z, e_1, e_2\rangle \to x'[y, q', c_1', c_2', e_1', e_2']$ when chosing a new transition. If the input head does not move, the scanned symbol is the same, G_m choses the new transition with $\langle x', q, Z, Z, e_1, e_2\rangle \to [x', q', c_1', c_2', e_1', e_2']$. The new transition now can be simulated in the same way as we have described above.

From the above explanations and the way of the construction of the components of Γ we can see that all successful computations of M correspond to a terminating derivation in Γ, and conversely, all terminating derivations in Γ correspond to a successful computation of M.

Now, if the recursively enumerable language L contains the empty word, $\varepsilon \in L$, we can add $S \to \varepsilon$ to the rules in P_m. This way the system also generates the empty word, with an erasing rule, of course. Hence, the result. □

References

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

[2] J. Dassow, Gh. Păun, G. Rozenberg, Grammar Systems. *Handbook of Formal Languages*, Vol. 2, Chapter 4, ed. by G. Rozenberg and A. Salomaa, Springer-Verlag, Berlin, 1997, 155-213.

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

[4] P. C. Fischer, Turing machines with restricted memory access, *Information and Control*, 9 (1966), 364-379.

[5] J. E. Hopcroft, J. D. Ullman, *Introduction to Automata theory, Languages and Computation,* Addison-Wesley, 1979.

[6] N. Mandache, On the computational power of context-free PCGS, submitted.

[7] Gh. Păun, Parallel communicating grammar systems: recent results, open problems, *Acta Cybernetica* 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] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.