On the Computational Complexity of Context-Free Parallel Communicating Grammar Systems[1]

Stefan Bruda

Romanian Academy of Sciences
Research Institute for Artificial Intelligence
13 Calea 13 Septembrie
Bucharest 050711, Romania
Email: bruda@racai.ro

Abstract. In this paper we investigate the computational complexity for Parallel Communicating Grammar Systems (PCGSs) whose components are context-free grammars. We show that languages generated by non-returning context-free PCGSs can be recognized by $O(n)$ space-bounded Turing machines. Also we state a sufficient condition for linear space complexity of returning context-free PCGSs. Based on this complexity characterization we also investigate the generative power of context-free PCGSs with respect to context-sensitive PCGSs and context-sensitive grammars.

1 Introduction

Parallel Communicating Grammar Systems (PCGSs) have been introduced as a language-theoretic treatment of multiagent systems [4]. A PCGS consists of several components (grammars) which work parallelly, in a synchronized manner. This is done according to the communicating protocol in which one grammar (component) may query strings generated by others and several components may make queries at the same time [4]. Formal definitions will be review in the next section. Because of the synchronization and communication facilities, PCGSs whose components are of a certain type are more powerful than a single Chomsky grammar of the same type [1, 4].

The study of computational complexity of PCGS is a stand alone problem. It is also a feasible approach toward the generative power of PCGSs. By proving the upper-bound or lower-bound complexity for PCGSs of a certain type, it is possible to find out the relationship between the generative power of such PCGSs and that of other generative devices.

[1] in G. Păun and A. Salomaa (eds.) *New Trends in Formal Languages*, Springer Lecture Notes in Computer Science 1218, 1997, pp. 256–266

In this paper, the study of the computational complexity of context-free PCGSs is based on space-bounded Turing machines. We will show that languages generated by non-returning context-free PCGSs can be recognized by nondeterministic Turing machines using $O(|w|)$ tape cells for each input instance w. This result is obtained for both centralized and non-centralized non-returning PCGSs and a sufficient condition for the returning case was stated. Starting from these results, we will analyze the generative power of context-free PCGSs according to the generative power of context-sensitive grammars and context-sensitive PCGSs.

The present paper is organized as follows. The next section reviews some fundamental concepts, the third section presents the computational complexity of context-free PCGSs and the fourth section discusses the generative power of this kind of PCGSs. We present some final remarks and further studies in the last section.

2 Fundamentals

In this section, we will briefly review the notion of PCGS and that of space-bounded Turing computation. More detailed descriptions can be found in [1] and [2] respectively. We also assume that the reader is familiar with basic concepts in formal language and computational complexity theories.

We will use the notations from [1]. For $x \in V^*$, and a set U, $|x|_U$ denotes the number of occurrences of elements of U in x. We also define $\|U\|$ to be the cardinality of set U. The null string is denoted by λ.

The next definitions are also conforming to [1].

Definition 2.1 Let $n \geq 1$ be a natural number. A PCGS with n components is a $(n+3)$-tuple

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

where N is the set of nonterminals, T is a terminal alphabet, $K = \{Q_1, Q_2, ..., Q_n\}$ (the sets N, K and T are mutually disjoint) and

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

are Chomsky grammars. Let $V_\Gamma = N \cup K \cup T$.

The grammars G_i, $1 \leq i \leq n$, are the components of the system and the elements of K are called query symbols; their indices points to $G_1, ..., G_n$ respectively.

The derivation in a PCGS id defined as follows.

Definition 2.2 Given a PCGS $\Gamma = (N, K, T, G_1, ..., G_n)$ as in the definition above, for the tuples $(x_1, x_2, ..., x_n)$, $(y_1, y_2, ..., y_n)$, $x_i, y_i \in V_\Gamma^*, 1 \leq i \leq n$, we write $(x_1, x_2, ..., x_n) \Rightarrow (y_1, y_2, ..., y_n)$ if one of the following cases holds:

1. $|x_i|_K = 0$, $1 \leq i \leq n$, and for all i, $1 \leq i \leq n$, we have $x_i \Rightarrow y_i$ in G_i or $x_i \in T^*$ and $x_i = y_i$;

2. there is i, $1 \leq i \leq n$, such that $|x_i|_K > 0$ and for each such i let $x_i = z_1 Q_{i_1} z_2 Q_{i_2} ... z_t Q_{i_t} z_{t+1}$, $t \geq 1$; in that case, for $z_j \in V_\Gamma^*$, $|z_j|_K = 0$, $1 \leq j \leq t+1$, if

$\left|x_{i_j}\right|_K = 0$, $1 \leq j \leq t$, then $y_i = z_1 x_{i_1} z_2 x_{i_2} ... z_t x_{i_t} z_{t+1}$ [and $y_{i_j} = S_{i_j}$, $1 \leq j \leq t$]. If exists j, $1 \leq j \leq t$, and $\left|x_{i_j}\right|_K \neq 0$ then $y_i = x_i$. For all i, $1 \leq i \leq n$, for which y_i was not specified above we have $y_i = x_i$.

The first case is called a componentwise derivation step and the second a communication step. Note that communications have priority over componentwise derivations. The query symbol to which a string has been communicated is called *satisfied*.

A tuple $(x_1, x_2, ..., x_n)$ is called a *configuration* of the system. We will call x_i a *component of the configuration* or only a *component* if the reference to the configuration is understood from the context.

Note that rules $Q_j \to \alpha$ are never used, so we can assume that there are no such rules [1].

The derivation in a PCGS is blocked if no rewriting rule can be applied to a nonterminal symbol in any component or circular queries appear (this happens when G_{i_1} introduces Q_{i_2}, G_{i_2} introduces Q_{i_3}, ..., $G_{i_{k-1}}$ introduces Q_{i_k} and G_{i_k} introduces Q_{i_1}. In this case no rewriting step is applicable, because the communication has priority, but also no communication steps are applicable).

Definition 2.3 The language generated by a PCGS Γ is:
$L(\Gamma) = \{x \in T^* | (S_1, S_2, ..., S_n) \Rightarrow^* (x, \alpha_2, ..., \alpha_n), \alpha_i \in V_\Gamma^*, 2 \leq i \leq n\}$.

The derivation starts from the tuple of axioms $(S_1, S_2, ..., S_n)$. A number of rewriting and/or communication steps are performed until G_1 produces a terminal string. Note that $L(\Gamma)$ contains only strings generated by the first component, with no care about the strings generated by the others, which may contain query symbols.

Definition 2.4 Let $\Gamma = (N, K, T, G_1, ..., G_n)$ be a PCGS. If only G_1 is allowed to introduce query symbols, then Γ is called *centralized*. The unrestricted case is called *non-centralized*.

Definition 2.5 A PCGS is called *returning* (to the axiom) if, after communication, a component which has communicated a string resumes the work from its axiom as described by sentence [and $y_{i_j} = S_{i_j}$, $1 \leq j \leq t$] in the second case of the definition 2.2. A PCGS is called *non-returning* if components continue working using the current string after a query (i.e. the sentence above is erased from the definition).

Notations: A centralized, returning PCGS with components of type X is denoted by PC_*X. For the centralized case we add a C and for non-returning case a N (see [1] for details). We obtain the classes PC, CPC, NPC, $NCPC$.

The notion of the *coverability tree* of a non-returning PCGS has been introduced in [5]. We will summarize here this notion and its relevant properties for this paper.

The set of natural numbers $\mathbb{N}$ is extended by a special symbol ω to the set $\mathbb{N}_\omega = \mathbb{N} \cup \{\omega\}$. The operations "+", "-", "$\cdot$" and the relation "$\leq$" over $\mathbb{N}$ are

extended to $\mathbb{N}_\omega$ by $\omega+\omega=\omega+n=n+\omega=\omega$, $\omega-n=\omega$, $\omega\cdot n=n\cdot\omega=\omega$, $n\leq\omega$ for all $n\in\mathbb{N}$.

The set N (of nonterminals) of a PCGS $\Gamma=(N,K,T,G_1,...,G_n)$ is ordered: $A_1,...,A_{n+m}$, $m\geq 0$, such that $A_1=S_1$, ..., $A_n=S_n$.

Let $w=(w_1,...,w_n)$ be a configuration of Γ. M_w denotes the vector

$M_w=((|w_1|_{X_1},...,|w_1|_{X_{2n+m}}),...,(|w_n|_{X_1},...,|w_n|_{X_{2n+m}}))$,

where $X_i=A_i$, $1\leq i\leq n+m$, $X_{n+m+j}=Q_j$, $1\leq j\leq n$. $M_w(i,j)$ denotes the element $|w_i|_{X_j}$.

We can assume [5] that for each component of Γ there is a phantom production which does not change the string and which can be applied only to terminal strings in the synchronized case. So, a rewriting step in Γ is a n-tuple $t=(r_1,...,r_n)$, where r_i denotes either a production in G_i or the phantom production, for all $1\leq i\leq n$. For uniformity, we say that communication steps are produced by a special transition Λ. The set of all $t=(r_1,...,r_n)$ as above is denoted by $TR(\Gamma)$, $\Lambda\in TR(\Gamma)$. A transition t is enabled in a certain configuration if the corresponding rewriting or communication step can be applied in that configuration.

If a transition t is enabled for a configuration w of Γ then we write $M_w[t>_\Gamma$; if, after t is performed, the new configuration is w' then we write $M_w[t>_\Gamma M_{w'}$.

Let A and B be two arbitrary sets. $\mathcal{T}(V,E,l_1,l_2)$ is an (A,B)-labeled tree if (V,E) is a tree and $l_1:V\to A$ is the node labelling function and $l_2:E\to B$ is the edge labelling function. We denote by $d_{\mathcal{T}}(v_1,v_2)$ the set of all nodes on the path from v_1 to v_2.

For each PCGS Γ there is a $((\mathbb{N}_\omega^{2n+m})^n,TR(\Gamma))$-labeled tree called *the coverability tree* for Γ defined as follow [5].

Definition 2.6 Let $\Gamma=(N,K,T,G_1,...,G_n)$ be a PCGS. A $((\mathbb{N}_\omega^{2n+m})^n,TR(\Gamma))$-labeled tree, $\mathcal{T}=(V,E,l_1,l_2)$, is called a coverability tree of Γ if the following hold:

1. the root, denoted by v_0, is labeled by M_{x_0}, where $x_0=(S_1,...,S_n)$ (the initial configuration);
2. for any node $v\in V$ the number of outgoing edges $|v^+|$ is
   - 0, if either there is not any transition enabled at $l_1(v)$ or there is $v'\in d_{\mathcal{T}}(v_0,v)$ such that $v\neq v'$ and $l_1(v)=l_1(v')$
   - the number of transitions enabled al $l_1(v)$ otherwise.
3. for any $v\in V$ with $|v^+|>0$ and any transition t which is enabled at $l_1(v)$ there is a node v' such that:
   (a) $(v,v')\in E$,
   (b) $l_2(v,v')=t$,
   (c) $l_1(v')$ is given by

- let M be such that $l_1(v)[t >_\Gamma M$;
- if M contains queries then $l_1(v') = M$ else
  if exists $v^* \in d_\mathcal{T}(v_0, v)$ such that $l_1(v^*) \leq M$ and $l_1(v^*)(i,j) < M(i,j)$
  then $l_1(v')(i,j) = \omega$ else
  $l_1(v')(i,j) = M(i,j)$,
  for all i, j, $1 \leq i \leq n$ and $1 \leq j \leq 2n+m$.

For non-returning synchronized PCGSs such tree is always finite and can be effectively constructed (see [5] for demonstrations and details). The coverability tree for a PCGS Γ is denoted by $\mathcal{T}(\Gamma)$.

From the construction of the coverability tree follows that if, for some configuration w, $M_w(i,j) = \omega$ then, in that configuration, the number of occurrences of X_j in the i-th component can be made arbitrarily large [5]. This implies that *if $M_w(i,j) = \omega$ then X_j cannot be totally removed by any successive derivation steps from x_i, i.e. such nonterminals cannot block the derivation.*

Definition 2.7 Given a Turing machine M and an input string $x \in T^*$, the working space of M on x is the length of work tapes for M to halt on x. More generally, let S be any function from $\mathbb{N}$ to $\mathbb{N}$; let $L \subseteq T^*$. We say that M decides [2] L in space S provided that M decides L and uses at most $S(n)$ tape cells on any input of length n in T^*. If M is a nondeterministic Turing machine we write $L \in NSPACE(S(n))$. We say also that M is a $S(n)$ space-bounded Turing machine.

For the rate of growth of a function we have the following definition [2]:

Definition 2.8 Let f and g be natural functions. We write $f = O(g)$ iff there is a constant $c > 0$ and an integer n_0 such that $f(n) \leq c\,g(n)$ for all $n \geq n_0$.

3 The Complexity of Context-Free PCGS

In this section we will study the computational complexity of PCGSs whose components are context-free grammars. We suppose there are not λ-productions. A discussion on λ-productions will be done at the end of this section.

Definition 3.1 During a derivation process in a PCGS, a component of the current configuration x_i is called *non-direct-significant* for the recognizing of the string w if

(i) either $i \neq 1$ and x_i is not queried anymore or

(ii) i=1 and the derivation from x_1 to w in G_1 cannot end successfully unless x_1 is reduced to the axiom sometime in the future or

(iii) $i \neq 1$ and x_i is queried by x_j, $j \neq i$, and x_j become non-direct-significant.

All the others components are called *direct-significant*. Any component which is reduced to the axiom becomes direct-significant.

[2] A Turing machine M decides a language L if, for any input string w, M halts and writes on its tape a specified symbol Y if $w \in L$ or another symbol N if $w \notin L$; If M writes Y we say it *accepts* the string, otherwise it *rejects* the input [2]

In other words, a non-direct-significant component of a PCGS cannot directly participate at a successful derivation. It can only produce lateral effects (by queries which can modify other components) or block the derivation (by circular queries or by its nonterminals for which there are no applicable rewriting rules).

This definition introduces the class of components for which the structure is irrelevant for the derivation. Therefore, these components can be erased if the information relevant for lateral effects is kept.

Starting from this definition we can consider the following lemmas.

Lemma 3.1 *Let $\Gamma = (N, K, T, G_1, ..., G_n)$ be a centralized PCGS ($\Gamma \in CPC_*CF \cup NCPC_*CF$) and $w \in T^*$ a string. Let also $(x_1, ..., x_n)$ be a configuration of the system. Then, if the length of a component x_i becomes greater than $|w|$, that component becomes non-direct-significant for the recognizing of w.*

Proof. We will consider two situations:

(i) let $i = 1$. If $|x_1|_K = 0$, then x_1 will be rewrited using the rules of G_1. But these are context-free rules and there are not λ-productions, so the length of x_1 does not decrease. If $|x_1|_K \neq 0$, a communication step will be performed. But the communication step does not reduce the length of the component because there are not null components to be queried (there are not λ-productions). So, the length of x_1 does not decrease anymore and this leads to the rejection of w because the first component is not queried (we have a centralized PCGS) so it can not be reduced to the axiom. Therefore x_1 is non-direct-significant according to the definition 3.1.

(ii) for $i \geq 2$, only the first component can introduce query symbols, so if x_i is queried by the first component (if x_i is not queried then it is obviously non-direct-significant), the length of x_1 becomes greater than $|w|$, therefore x_1 becomes non-direct-significant (according to the point (i)). So x_i is non-direct-significant. □

Lemma 3.2 *Let $\Gamma = (N, K, T, G_1, ..., G_n)$ be a non-centralized non-returning PCGS ($\Gamma \in NPC_*CF$) and $w \in T^*$ a string. Let also $(x_1, ..., x_n)$ be a configuration of the system. Then, if the length of a component x_i becomes greater than $|w|$, that component becomes non-direct-significant for the recognizing of w.*

Proof. The proof is basically similar to the proof of the lemma 3.1. The case $i = 1$ has the same proof as the case (i) in the proof above, because x_1 can not decrease even if it is queried (the system is non-returning).

For $i \geq 2$, either the component x_i is never queried therefore it is non-direct-significant, or it is queried by the first component and we have the same situation as in the case (ii) of the proof above, or it is queried by another component x_j, $j \neq i$, $j \neq 1$, which become in that way longer than w and also can not decrease. □

Lemma 3.3 *Let $\Gamma = (N, K, T, G_1, ..., G_n)$ be a non-centralized returning PCGS ($\Gamma \in PC_*CF$) and $w \in T^*$ a string. Let also $(x_1, ..., x_n)$ be a configuration of the system. Then, if the length of a component x_i becomes greater than $|w|$, that component becomes non-direct-significant for the recognizing of w.*

Proof. We have the same proof as for lemma 3.2 with the mention that, if a component is queried, it is reduced to the axiom and then it become direct-significant. But this situation is allowed by the definition (a non-direct-significant component can become direct-significant iff it is reduced to the axiom). □

Using the lemmas above, the complexity of context-free PCGS can be studied. We first consider the non-returning case.

Lemma 3.4 *Let Γ be a non-returning PCGS with n context-free components ($n \geq 1$). Then there is a Turing machine M that recognizes the language $L(\Gamma)$ using at most $O(|w|)$ amount of work tape space for each input instance w.*

Proof. Let $\Gamma = (N, K, T, G_1, ..., G_n)$ be a non-returning PCGS, where $G_i = (N \cup K, T, P_i, S_i)$, $1 \leq i \leq n$, are context-free grammars. We will construct the nondeterministic Turing machine M which recognizes $L(\Gamma)$.

M will be a standard Turing machine, with a work tape equipped with a read/write-head. The alphabet of the tape of M is $N \cup K \cup T \cup \{@, \omega\}$, $@, \omega \notin N \cup K \cup T$. Given an input string $w \in T^*$, M will simulate step by step the derivation of w by Γ. But first M should compute the coverability tree $\mathcal{T}(\Gamma)$ of Γ. Note that this computation can be done [5] and its space complexity is not w-dependent, so it does not modify the space complexity of the whole computation if this complexity is a function of w. Then M find the number $m_{max} = max\{l_1(v)(i,j) | 1 \leq i \leq n, 1 \leq j \leq 2n+m, l_1(v)(i,j) \neq \omega\}$. After that, M erases the coverability tree and keeps on its tape the number m_{max} [3].

The simulation of the derivation is done according to the definition 2.2. Therefore, there are two types of derivation steps to simulate: the componentwise rewriting and the communication. M will keep on its tape the current configuration and will work on it as follows:

(i) If $|x_i|_K = 0$ for all i, $1 \leq i \leq n$, M simulate rewriting for each component x_i, $1 \leq i \leq n$. If $|x_i|_N = 0$, then x_i remains unchanged. Otherwise, M nondeterminisicaly selects a rule from the rule set P_i and rewrites x_i according to this rule. If there are some i for which such rule does not exist, M rejects the input and halts.

If $|x_i| > |w|$ then, according with lemma 3.1 (if we have a centralized PCGS) or 3.2 (if the system is non-centralized), x_i become non-direct-significant. Therefore its structure is irrelevant and it will be replaced by the string

[3] Or, m_{max} being a property of Γ and not of w, it can be considered a parameter of M. Therefore it should not be computed (and so M does not need to compute $\mathcal{T}(\Gamma)$) but it should be on the tape at the beginning of the computation

$$@t_1T_1...t_jT_jq_1Q_1...q_kQ_k \qquad (1)$$

where @ is a special symbol ($@ \notin N \cup T \cup K$), T_1, ..., T_j are the distinct nonterminals in x_i and Q_1, ..., Q_k are the distinct query symbols in x_i. t_h ($1 \leq h \leq j$) is either the number of occurrences of the nonterminal T_h in x_i if these occurrences are few than m_{max} or ω otherwise. Also q_h ($1 \leq h \leq k$) is either the number of occurrences of the query symbol Q_h in x_i if these occurrences are few than m_{max} or ω otherwise.

Note than if the number of occurrences of X in x_i ,$X \in K \cup N$, becomes greater than m_{max}, then $l_1(v)$ must contain ω in the position corresponding to x_i and X (where v is the node in $\mathcal{T}(\Gamma)$ corresponding to the current configuration), so the number of occurrences of X cannot decrease (in fact it can grow indefinitely), therefore X cannot be eliminated from x_i, so it is not necessary to count its occurrences in x_i anymore.

We have to explain now how the rewriting works on strings of the form (1). Let the rewriting rule be

$A \to \alpha_1 A_1 \alpha_2 A_2 ... \alpha_m A_m \alpha_{m+1}$

where $A \in N$, $A_1, ..., A_m \in N \cup K$ and $\alpha_1, ..., \alpha_m \in T^*$. Then, there is $T_r = A$, $1 \leq r \leq j$, (if not, the rule is not applicable) and M increases the counter for each nonterminals or query symbol A_j (if that counter is ω then it remains unchanged), $1 \leq j \leq m$, in x_i; if that counter becomes greater than m_{max},then it is replaced by ω and if A_j does not already exists in x_i, then a new pair $1A_j$is added to x_i. Finally, M decrements the counter of A, excepting when this counter is ω, when it remains unchanged. If that counter becomes zero, both this counter and A are erased from x_i.

The reason for keeping nonterminals in non-direct-significant components is that these nonterminals can introduce query symbols when a rewriting is performed. Also the absence of a nonterminal can block the derivation.

(ii) If there are query symbols in the current configuration, then M simulates communication steps. If there are circular queries, M rejects the input and halts. Otherwise, M nondeterministicaly selects a component x_i for which Q_j, $1 \leq j \leq q$, are all the query symbols and $|x_j|_K = 0$, $1 \leq j \leq q$. M sequentially replaces Q_j by x_j. If either the current x_j is of the form (1) or, after replacement, x_i becomes longer than $|w|$, then x_i becomes non-direct-significant, so it will be replaced by a string of the form (1).

This communication step is repeatedly performed until there are no query symbols in the current configuration.

M repeats steps of type (i) and (ii) until:

1. either x_1 and w are identical or
2. the first symbol of x_1 is @ or
3. the number of iterations exceeds a fixed positive number c.

In the first case M accepts the input and halts, in the other two cases M rejects w and halts.

Let us count the amount of work space used by M during the derivation. If the length of a component x_i is less than $|w|$ then this component is kept on the tape as it is, so less than $|w|$ tape cells are necessary in order to keep it. If a component has a length greater than $|w|$, it become of the form (1). Because we have a fixed finite number t of nonterminals for a given PCGS and exactly n query symbols, the length of such component on the tape is independent of $|w|$ and is less than $1 + \log m_{max}(t + n)$, where $t = \|N\|$.

A communication step may use temporary an amount of tape space double than the space used by a single component (e.g. a string of length $|w|$ is queried by another string of length $|w|$; before the reduction to form (1) we have to use $2|w|$ tape cells).

Therefore, the number of cells used by a component is less than

$2max(|w|, 1 + \log m_{max}(t + n))$.

We have n components and we need some extra space on the tape to keep the rules of the system and m_{max}. So, the space used by M is upper-bounded by

$2n \cdot max(|w|, 1 + \log m_{max}(t + n)) + pl + \log m_{max}$.

But t, n and m_{max} are not $|w|$-dependent so, conforming to the definition 2.8 of the rate of growth of the functions, the space used is

$O(2n|w| + pl + \log m_{max})$.

Finally we have to show that the fixed integer c we claimed before exists. But this is immediate from the following property of space-bounded Turing machines [2]:

$NSPACE(S) \subseteq \cup\{NTIME(d^S)|d \geq 1\}$

If the number of the iterations of M becomes greater than c, M have repeated some configurations, so M should reject the input because, if the input is in $L(\Gamma)$, it would have been accepted before the configuration is repeated at the second time (this happens because of the nondeterminism of M). □

The same result cannot be obtained for returning PCGS unless there is a limit for the number of significant occurrences for each nonterminal (i.e. if the number of occurrences of any nonterminal in any component x_i of the configuration exceeds that limit, then that nonterminal cannot be eliminated from x_i by any further derivation). We will call this limit *a limit of significant occurrences*.

Note that this limit was found for the non-returning case by constructing and inspecting the coverability tree of the system in discussion. This is possible because this tree can be effectively constructed for the non-returning case [5]. The construction of the coverability tree is not necessary effective for returning PCGSs.

Lemma 3.5 *Let Γ be a returning centralized PCGS with n context-free components, $(n \geq 1)$. Then there is a Turing machine M that recognizes the language $L(\Gamma)$ using at most $O(|w|)$ amount of work tape space for each input instance w if there is a finite limit $m_{max} = m(|w|)$ of significant occurrences for any nonterminal, where $m : I\!N \to I\!N$, $m = O(d^n)$, $d > 1$.*

Proof. Let $\Gamma = (N, K, T, G_1, ..., G_n)$ be a non-returning PCGS, where $G_i = (N \cup K, T, P_i, S_i)$, $1 \leq i \leq n$, are context-free grammars. We will construct the

nondeterministic Turing machine M which recognizes $L(\Gamma)$.

M will be a standard Turing machine, with a work tape equipped with a read/write-head. The alphabet of the tape of M is $N \cup K \cup T \cup \{@, \omega, \}$. Given an input string $w \in T^*$, M will simulate step by step the derivation of w by Γ. The construction of M is basically similar to the one used in the proof of lemma 3.4 excepting that M does not compute the coverability tree of Γ.

The reference to lemma 3.2 from the above demonstration should be replaced in the current demonstration by the reference to lemma 3.3.

Differently from the non-returning case, when a component x_i is queried, M has to simulate the returning of x_i to the axiom. This is done by replacing x_i by the axiom of its grammar (S_i).

Note that this replacement does not depend of the form of x_i so the processing of strings longer than $|w|$ is correct, i.e. the rewriting of such components in the form (1) does not lose any necessary information. Moreover, a number of occurrences (of any terminal X in any component of the configuration x_i) greater than m_{max} implies that X cannot be eliminated from x_i, as in the proof of lemma 3.4. Therefore, the non-direct-significant components are correctly stored.

M halts if

1. x_1 is identical with w; in this case M accepts the input or

2. no derivation steps are available (there are not rules applicable for some components or there are circular queries) and M rejects the input or

3. the number of iterations exceeds a fixed positive number (similar with the one in lemma 3.4); in this case also M rejects the input.

Even if the circularity of a PCGS is not a decidable problem for the returning case, M halts in any situation because of the limit c of its possible configurations. M does not decide the circularity of the system at the beginning of the derivation (which can be an undecidable problem) but it halts when any circularity appears.

Finally, the space used by M is, analogous with the proof of lemma 3.4,

$O(2n \cdot max(|w|, 1 + \log m_{max}(t + n)) + pl + \log m_{max}) =$

$= O(2n \cdot max(|w|, 1 + \log m(|w|)(t + n)) + pl + \log m_{max}) =$

$= O(2n \cdot max(|w|, 1 + \log d^{|w|})(t + n)) + pl + \log d^{|w|}) =$

$= O(max(|w|, |w|) + |w|) = O(|w|)$

(because $m_{max} = O(2^{|w|})$), and a limit c for the possible configurations of the tape can be found. □

By lemmas 3.4 and 3.5 we have

Theorem 3.1 *$\mathcal{L}(X_*CF) \subseteq NSPACE(n)$, $X \in \{NPC, NCPC\}$ and there are no λ-productions.*

Theorem 3.2 *$\mathcal{L}(X_*CF) \subseteq NSPACE(n)$ ($X \in \{PC, CPC\}$ and there are no λ-productions) if a limit in $O(d^{|w|})$, $d > 1$, of significant occurrences exists.*

Also we can consider a subclass of context-free PCGS with λ-productions. This subclass is very restrictive but we can consider in this way PCGSs which can generate the null string.

Definition 3.2 $\Gamma \in X_*CF_{\lambda^*}$, $X \in \{PC, NPC, CPC, NCPC\}$, if $\Gamma \in X_*CF$, X as above, and either Γ does not contain λ-productions or

(i) P_i, $i > 1$, do not contain λ-productions and

(ii) P_1 contains only the three productions $S_1 \to \lambda$, $S_1 \to S_1$ and $S_1 \to Q_2$ and

(iii) x_1 is not queried anymore (i.e. Q_1 does not appear in the right side of any production of the system).

Note than the subclasses introduced by this definition are similar with usual context-free grammars in which λ-productions are eliminated [2].

We have the following theorem.

Theorem 3.3 $\mathcal{L}(X_*CF_{\lambda^*}) \subseteq NSPACE(n)$, $X \in \{NPC, NCPC\}$.

Proof. Let $\Gamma = (N, K, T, G_1, ..., G_n)$. We will construct Turing machines which simulate the derivation of an input string w. Such machine M works as follows:

If w is a null string, which belongs to the language in discussion, M accepts it and halts. Otherwise, M continues the derivation for the system $\Gamma' = (N, K, T, G_2, ..., G_n)$ as the machine for the appropriate class X_*CF does. Note that, by erasing the first component, the system becomes without λ-productions, so the machine works properly. Also, the first component in Γ only waits for the second component to obtain a terminal string and queries it. □

Corollary 3.1 $\mathcal{L}(X_*CF_{\lambda^*}) \subseteq NSPACE(n)$, $X \in \{PC, CPC\}$ *if a limit in* $O(d^{|w|})$, $d > 1$, *of significant occurrences exists.*

4 Generative Power of Context-Free PCGS

In this section we will analyze the generative power of context-free PCGS with respect to context-sensitive grammars but also to other types of PCGSs.

Theorem 4.1 $\mathcal{L}(X_*CF) \subseteq \mathcal{L}(CS)$, $X \in \{NPC, NCPC\}$.

Proof. It has been proved that the class of languages recognized by linear space-bounded Turing machines is identical to the class of context-sensitive languages [2]. This and theorem 3.1 imply that $\mathcal{L}(CS)$ includes $\mathcal{L}(X_*CF)$. □

Corollary 4.1 $\mathcal{L}(X_*CF) \subseteq \mathcal{L}(Y_*CS)$, $X \in \{NPC, NCPC\}$, $Y \in \{NPC, NCPC\}$.

We have proved so that any language generated by non-returning context-free PCGSs is context-sensitive. An open problem is if there are context-sensitive languages which can not be generated by such PCGS, i.e. if the inclusion in the theorem 4.1 is proper.

The above results are obtained for the classes X_*CF ($X \in \{NPC, NCPC\}$) but they can be extended for $X \in \{PC, CPC\}$ if a limit of significant occurrences as above exists. Also these results are true for the classes $X_*CF_{\lambda^*}$ (X as above).

5 Conclusions

In this paper we have tried to investigate the computational complexity of context-free PCGSs. We have proved the linear space complexity of languages generated by non-returning context-free PCGS, proving so that these languages are context-sensitive. Also, we have found some results concerning returning systems. Finding the limit of significant occurrences we mentioned above is an open problem which we are working on.

Systems which contains λ-productions were not considered and this is a possible extension of this study. We think a feasible approach to this problem consists in finding some transformations which eliminates λ-productions (in the same manner as for context-free grammars [2]) even if there are synchronization problems. The theorem 3.3 is a support for this approach. We believe that these systems have linear space complexity too.

Also a possible extension of this study is the investigation of time-bounded complexity of context-free PCGSs. We intend to pursue further studies on these issues.

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] H. R. Lewis, C. H. Papadimitriou, *Elements of the Theory of Computation*, Prentice-Hall, 1981;

[3] L. Cai, *The Computational Complexity of Linear PCGS*, Computer and AI, 15, 2 - 3 (1996), 199 - 210;

[4] Gh. Păun, L. Sântean, *PCGS: The Regular Case*, Ann. Univ. Buc., Matem. Inform. Series, 38, 2(1989), 55 - 63;

[5] F. L. Ţiplea, O. Procopiuc, C. M. Procopiuc, C. Ene, *On the Power and Complexity of PCGS*, Artificial Life: Grammatical Models, Black Sea University Press, Bucharest, 1995.