# CDP GRAMMAR SYSTEMS – A NEW MODEL

**Sindhu J Kumaar[1] and P. J. Abisha[2]**

[1] Department of Mathematics, B. S. Abdur Rahman University
Chennai – 600048, India
[2] Department of Mathematics, Madras Christian College
Tambaram, Chennai –600059, India

**Abstract**

Motivated by pattern grammars of Dassow et al [3] and cooperating distributed grammar systems by Csuhaj – Varju et al [5], we introduce a new grammar system, called cooperating distributed pattern grammar system CDPGS. In this system all the components considered are Pattern grammars. The resultant family of languages is compared with other families of languages and also we introduce a learning algorithm for cooperating distributed pattern grammar system.

***Keywords: Pattern grammar, Cooperating Distributed grammar system, Learning, Oracle.***

## 1. Introduction

Angluin et al [1], while studying the problem of learning or inferring a pattern common to all strings in a given sample, introduced pattern languages. A pattern is a finite string of constants (or terminal symbols) and variables (or non terminal symbols). A pattern language is the set of all strings obtained by substituting for each variable symbol in the pattern, any nonempty strings of constants, with different occurrences of the same variable being replaced by the same string.

Dassow et al [3], defined a new generative device, called a pattern grammar, motivated by the study of Angluin [1]. The idea here is to start from a finite set A of axioms which are over an alphabet of constants; given a set P of patterns which are strings over constants and variables, replace the variables in a given pattern by axioms and continue the process with the current set of strings, obtained by such operations. The replacement is done in parallel which means all variables occurring in a pattern are replaced simultaneously. It is also uniform which means same variables are replaced by same string at a particular step. All strings generated in this way constitute the associated language called a pattern language.

The theory of grammar systems [11] is an interesting and a deeply investigated area of formal language theory. A variety of grammar systems have been considered in the literature and among these a cooperating distributed (CD) grammar system by Csuhj – Varju et al [5] has been studied by many researchers. A compact account of many of these details is provided by Dassow et al [2]. A cooperating distributed (CD) grammar system comes from black board systems. Each component grammar corresponds in this model to a particular knowledge source of the black board system. The global data base of the black board systems – the black board – is modeled by sentential form in which the component grammars of the grammar system make their rewritings. Here we examine CD grammar systems whose components are pattern grammars

## 2. Pattern Grammar

A pattern with k variables is a word over T ∪ X, where X = {$x_1, x_2, \ldots, x_k$}. A pattern with k variables is said to be in canonical form if, for each i ≤ k, the right most occurrence of $x_i$ in p occurs to the right of the rightmost occurrence of $x_{i-1.}$ For a pattern p with k variables and a set of k strings $u_1, u_2, \ldots u_k \in T^*$, let $p[x_{1:}u_1, x_{2:}u_2, \ldots x_k : u_k]$ denote the strings obtained by substituting $u_i$ for each occurrence of $x_i$ in p. The language L(p) = { $p[x_{1:}u_1, x_{2:}u_2, \ldots x_k : u_k]$ / $u_1, u_2, \ldots u_k \in T^*$} generated by using substitutions of this type is the language generated by the pattern p. We first recall the definition of pattern grammar by Dassow et al [8].

**Definition 2.1:** [8] A pattern grammar (PG) is a 4 – tuple $G = (\sum, X, A, P)$ where ∑ is an alphabet whose elements are called constants, X is an alphabet whose elements are called variables. A ⊆ $\sum^*$ is a finite set of elements of $\sum^*$ called axioms and P ⊆ $(\Sigma \cup X)^+$ is a finite set of words called patterns where each word contains atleast one variable. The rewriting is done as follows:

$$P(A) = \left\{ \begin{matrix} u_1\, x_{i_1}\, u_2\, x_{i_2} \ldots u_k\, x_{i_k}\, u_{k+1} / u_1\, \delta_{i_1}\, u_2\, \delta_{i_2} \ldots u_k\, \delta_{i_k}\, u_{k+1} \in P, \\ u_i \in \sum^*, 1 \le i \le k+1,\ x_{i_k} \in A,\ \delta_{i_k} \in X, 1 \le i \le k \end{matrix} \right\}$$

This means that, P(A) contains words obtained by replacing the variables in the pattern by words from A and different occurrences of the same variable are replaced by the same word. Then the pattern language (PL) generated by G is $L(G) = P \cup A \cup P(A) \cup P(P(A)) \cup \ldots$

**Example 2.1:** $G = (\{a, b\}, \{\delta\}, \{ab\}, \{a\delta b\})$ is a pattern grammar generating language $L(G) = \{a^n b^n / n \geq 1\}$, as $A = \{ab\}$, $P(A) = \{aabb\}$, $P(P(A)) = \{aaabbb\}$ and so on.

We observe that the concept of variables is similar to that of variable in Chomskian grammar. But the rewriting process is different; it is uniform, in the sense that all the occurrences of a variable in a pattern are replaced by the same word and the variables are rewritten in parallel. Still, the pattern grammars generate languages which are incomparable with Chomskian languages and Lyndenmayer languages.

## 3. Cooperating Distributed Grammar System

**Definition 3.1:** [5] A cooperating distributed grammar system (CD grammar system for short) is an (n + 2) tuple $\Gamma = (T, G_1, G_2, \ldots, G_n, S)$
where,

- For $1 \leq i \leq n$, each $G_i = (N_i, T_i, P_i)$ is a (usual) context – free grammar with the set $N_i$ of non terminals, the set $T_i$ of terminals, the set $P_i$ of context – free rules, and without axiom,
- T is a subset of $\bigcup_{i=1}^{n} T_i$ ,
- $S \in \bigcup_{i=1}^{n} N_i = N$

The grammars $G_i$, $1 \leq i \leq n$, are called the components of $\Gamma$. Further $V_i = N_i \cup T_i$ and $V_\Gamma = \bigcup_{i=1}^{n} V_i$

The grammars correspond to the agents solving the problem on the black board; any rule represents some pieces of knowledge which results in a possible change on the black board. The axiom S is the formal counterpart of the problem on the black board in the beginning. The alphabet T contains the letters which correspond to such knowledge pieces which are accepted as solutions / part of solutions.

**Definition 3.2:** Let $\Gamma$ be a CD grammar system as in Definition 3.1. Let $x, y \in V_i^*$. Then we write $x \Rightarrow^k_G y$ iff there are words $x_1, x_2 \ldots\ldots x_{k+1}$ such that

(i) $x = x_1$, $y = x_{k+1}$,
(ii) $x_j \Rightarrow_G x_{j+1}$, ie., $x_j = x' A_j x_j''$, $x_{j+1} = x' w_j x_j''$, $A_j \rightarrow w_j \in P_i$, $1 \leq j \leq k$

Moreover, we write

$$x \Rightarrow^{\leq k}_{G_i} y \quad \textit{iff } x \Rightarrow^{k'}_{G_i} y \textit{ for some } k' \leq k,$$

$$x \Rightarrow^{\geq k}_{G_i} y \quad \textit{iff } x \Rightarrow^{k'}_{G_i} y \textit{ for some } k' \geq k,$$

$$x \Rightarrow^{*}_{G_i} y \quad \textit{iff } x \Rightarrow^{k}_{G_i} y \textit{ for some } k,$$

$$x \Rightarrow^{t}_{G_i} y \quad \textit{iff } x \Rightarrow^{*}_{G_i} y \textit{ and there is no } z \neq y \textit{ with } y \Rightarrow^{*}_{G_i} z.$$

Any derivation $x \Rightarrow^{k}_{G_i}$ y corresponds to k discrete derivation steps in succession in the grammar $G_i$, and this represents k changes of the partial solution on the blackboard by one of the agents according to her/his/its rules reflecting the knowledge. Thus the $\leq$ k-derivation mode corresponds to a time limitation, since the agent can perform at most k changes. The $\geq$ k-derivation mode represents competence since it requires that the agent can perform at least k changes, she/he/it must contribute at least k times in succession to the solving. The *-mode meets the case where the agent can work at the blackboard as long as she/he/it wants to do. Finally, the t-mode of derivation corresponds to that strategy where any agent has to perform solving steps at the blackboard as long as she/he/it can contribute to the process of solving.

**Definition 3.3:** Let
$f \in \{t, *, 1, 2, \ldots, \leq 1, \leq 2, \ldots, \geq 1, \geq 2, \ldots\}$,
and let $\Gamma$ be a CD grammar system. Then the language $L_f(\Gamma)$ generated by $\Gamma$ is defined as the set of all words $z \in T^*$ for which there is a derivation

$$S = w_0 \Rightarrow^{f}_{G_{i_1}} w_1 \Rightarrow^{f}_{G_{i_2}} w_2 \Rightarrow^{f}_{G_{i3}} \ \ldots \ \Rightarrow^{f}_{G_{ir}} w_r = z.$$

We now give some examples in order to illustrate the concepts.

**Example 3.1:** Let us consider the CD grammar system
$\Gamma = (\{a, b, c\}, G_1, G_2, S)$,
With
$G_1 = (\{A, B\}, \{A', B', a, b, c\}, \{A \rightarrow aA'b, B \rightarrow cB', A \rightarrow ab, B \rightarrow c\})$,
$G_2 = (\{S, S', A', B'\}, \{A, B\}, \{S \rightarrow S', S' \rightarrow AB, A' \rightarrow A, B' \rightarrow B\})$
Then we obtain
$L_1(\Gamma) = L_*(\Gamma) = L_{\leq k}(\Gamma) = L_{\geq 1}(\Gamma) = L_t(\Gamma) = \{a^n b^n c^m | n \geq 1, m \geq 1\}$, $k \geq 1$,
$L_2(\Gamma) = L_{\geq 2}(\Gamma) = L_t(\Gamma) = \{a^n b^n c^n | n \geq 1, n \geq 1\}$,
$L_k(\Gamma) = L_{\geq k}(\Gamma) = 0$ for $k \geq 3$.
We show the relation only for $L_i(\Gamma)$ and $L_2(\Gamma)$; the proofs for the other relations are analogous and left to the reader.

It is clear that the derivation starts with $S \Rightarrow S' \Rightarrow A\ B$ using the rules from $P_2$. We now have to change to $P_1$ (or the derivation is already blocked in the k- or $\geq$k-mode of derivation for $k \geq 3$) and one of the following cases holds:

*Case 0.* $A\ B \Rightarrow^2 abc.$
*Case 1.* $A\ B \Rightarrow^2 abcB'.$
*Case 2.* $A\ B \Rightarrow^2 aA'bc.$

*Case 3.* $AB \Rightarrow^2 aA'bcB'$.
In Case 0 the derivation has been terminated.
First we consider the derivation in the t-mode.
In Case 1 we can continue only in the following way:
$abcB' \Rightarrow^t abcB \Rightarrow^t abccB' \Rightarrow^t abccB \Rightarrow^t abc^3B' \Rightarrow^t abc^3B \Rightarrow^t \ldots$
$\ldots \Rightarrow^t abc^{r-2}B \Rightarrow^t abc^{r-1}B' \Rightarrow^t abc^{r-1}B \Rightarrow^t abc^r$.
Analogously, in Case 2 only words and all words of the form $a^rb^rc$, $r \geq 2$, can be generated.

In Case 3 we have to continue with applications of $A' \to A$, $B' \to B$. This yields aAbcB. Using rules of $P_1$ we obtain $a^2b^2c^2$ or $a^2b^2c^2B'$ or $a^2A'b^2c^2$ or $a^2A'b^2c^2B'$. In all these cases we derive a word of the same structure as in the previous cases, only the power of the terminal letters is changed from 1 to 2. Therefore it is easy to see that

$$L_t(\Gamma) = \{a^nb^nc^m / n \geq 1, m \geq 1\}$$

Now let us consider the derivation in 2 – mode. Then in the above cases 1 and 2 the derivation is blocked since we can apply one rule of $P_2$. Thus the only correct derivations are of the form
$S \Rightarrow^2 AB \Rightarrow^2 aA'bcB' \Rightarrow^2 aAbcB \Rightarrow^2 a^2A'b^2c^2B' \Rightarrow^2 a^2Ab^2c^2B \Rightarrow^2 \ldots$
$\ldots \Rightarrow^2 a^{n-1}Ab^{n-1}c^{n-1}B \Rightarrow^2 abc^{r-1}B' \Rightarrow^t a^nb^nc^n$.
This proves

$$L_2(\Gamma) = \{a^nb^nc^n / n \geq 1\}$$

## 4. Cooperating Distributed Pattern Grammar System

In the new model we consider pattern grammars as components. We define the system with pattern grammars as components.

**Definition 4.1:** A cooperating distributed pattern grammar system (CDPGS for short) is an (n + 3) tuple

$$\Gamma = (\Sigma, G_1, G_2, \ldots, G_n, X, A)$$

where,
for $1 \leq i \leq n$, each $G_i = (\sum_i, X_i, P_i)$ is a (usual) pattern grammar where $\sum_I \subseteq \sum$ is an alphabet whose elements are called constants, $X_i \subseteq X$ is an alphabet whose elements are called variables. $A \subseteq \sum^*$ is a finite set of elements called axioms and $P_i \subseteq (\Sigma_i \cup X_i)^+$ is a finite set of words called patterns where each word contains atleast one variable. The grammar $G_i$, $1 \leq i \leq n$, are called the components of $\Gamma$. The rewriting in the ith component is done as follows: If L is the set of words given as input to the ith component then, in a single step, it gets the language $P_i(L)$ from $P_i$, where

$$P_i(L) = \left\{ \begin{matrix} u_1 x_{i_1} u_2 x_{i_2} \ldots u_k x_{i_k} u_{k+1} / u_1 \delta_{i_1} u_2 \delta_{i_2} \ldots u_k \delta_{i_k} u_{k+1} \in P_i, \\ u_i \in \sum^*, 1 \leq i \leq k+1,\ x_{i_k} \in L,\ \delta_{i_k} \in X_i,\ 1 \leq i \leq k \end{matrix} \right\}$$

This means that, $P_i(L)$ contains words obtained by replacing the variables in the pattern by words from L and different occurrences of the same variable are replaced by the same word. If ith component works for k steps continuously, then we write $y \in P_i^k(x)$ if there are words $x_1, x_2 \ldots\ldots x_{k+1}$ such that
(i) $x = x_1$, $y = x_{k+1}$,
(ii) $x_{j+1} \in P_i(\{x_j\})$, j = 1, 2, …, k
Moreover, we write
$y \in P_i^*(x)$ iff $y \in P_i^k(x)$ for some k.
The *-mode is the case where the agent can work at the blackboard as long as she/he/it wants to do.
In this section we give few examples for the model CDPGS. In CDPGS we start derivation from the first component and words that are collected in the last component form the required language.

**Example 4.1:** $\Gamma = (\{a, b, c\}, G_1, G_2, \{\delta_1, \delta_2\}, \{c\})$,
$G_1 = (\{a\}, \{\delta_1\}, \{\delta_1 b\})$
$G_2 = (\{b\}, \{\delta\}, \{a\delta_2\})$,
For k = 1, $L(\Gamma) = \{a^ncb^n / n \geq 1\}$.
The derivation is as follows: At the beginning the first component produces a word 'cb', and then this word is taken as the axiom for the second component and the word 'acb' is produced. Now the word 'acb' which is produced in second component is considered as axiom for the first component thus the word '$acb^2$' is produced, this is treated as axiom for the next derivation in the second component and the word $a^2cb^2$ is generated. After n subsequent derivations the language generated in the first component is $a^ncb^n$.

**Example 4.2:** $\Gamma = (\{a, b\}, G_1, G_2, \{\delta_1, \delta_2\}, \{\lambda\})$
$G_1 = (\{a\}, \{\delta_1\}, \{a\delta_1\})$
$G_2 = (\{b\}, \{\delta_2\}, \{\delta_2 b\})$,
If k = 1, then $L(\Gamma) = \{a^nb^n / n \geq 1\}$.
The derivation is as follows: At the beginning the first component produces a word 'a', and then this word is taken as the axiom for the second component and the word 'ab' is produced. Now the word 'ab' which is produced in second component is considered as axiom for the first component thus the word '$a^2b$' is produced, this is treated as axiom for the next derivation in the second component and the word $a^2b^2$ is generated. After n subsequent derivations the language generated in the second component is $a^nb^n$.
Similarly if k = 2, the language generated is $\{a^{2n}b^{2n} / n \geq 1\}$. And hence in general if there are k discrete derivation

steps in succession in the grammar $G_i$, the language generated is $\{a^{k\,n} b^{k\,n} / n \geq 1\}$.

**Example 4.3:** $\Gamma = (\{a, b, c\}, G_1, G_2, \{\delta_1, \delta_2\}, \{c\})$,
$G_1 = (\{a, b, c\}, \{\delta_1\},\{ a\delta_1 a, b\delta_1 b\})$
$G_2 = (\{a, b, c\}, \{\delta_2\}, \{ a\delta_2 a, b\delta_2 b\})$,
$L(\Gamma) = \{wcw^R / w \in \{a, b\}^+\}$.
The derivation is as follows: At the beginning the first component produces a words 'aca' and 'bcb', and then these words are taken as the axiom set for the second component and the words '$a^2c\ a^2$', 'bacab', 'abcba' and '$b^2cb^2$' are produced. Now this set of words are taken as axiom set for the first component and thus the words '$a^3c\ a^3$', 'abacaba', '$a^2bcba^2$', '$ab^2cb^2a$', '$ba^2ca^2b$', '$b^2acab^2$', 'babcbab' and '$b^3c\ b^3$' are generated. After n subsequent derivations the language generated is $\{wcw^R / w € \{a, b\}^+\}$

**Example 4.4:** $\Gamma = (\{a, b\}, G_1, G_2, \{\delta_1, \delta_2\}, \{b\})$,
$G_1 = (\{a\}, \{\delta_1\}, \{a\delta_1\})$
$G_2 = (\{b\}, \{\delta\}, \{\delta_2 b\})$,
$L(\Gamma) = \{a^n b^m / n, m \geq 1\}$.
If there is no restriction on k, then the language generated is $\{a^n b^m / n, m \geq 1\}$.

**Proposition 4.1**: $CD(PL) - PL \neq \phi$
**Proof:** This is seen from the fact that the language $\{a^n b^m / n, m \geq 1\}$ is in Cooperating Distributed Pattern Language CDPL but not in pattern language PL.

**Proposition 4.2**: The family of Cooperating Distributed Pattern languages CDPL and the family of Pattern languages PL are comparable
**Proof:** The language $\{a^n b^n / n \geq 1\}$ is in PL for it is generated by the pattern grammar $(\{a, b\}, \{\delta\}, \{ab\}, \{a\delta b\})$ also it is in CDPL. Thus $CDPL \cap PL \neq \phi$

**Proposition 4.3**:
1. The family of context – free languages (CFL) and CDPL are comparable
2. The family of regular languages (RL) and CDPL are incomparable

**Proof:**
1 The statement 1 is due to the fact that the language $\{wcw^R / w \in \{a, b\}^+\}$ is in both CFL and CDPL
2 The statement 2 is due to the fact that the language $\{a^n b^n / n \geq 1\}$ is in CDPL but not in RL

**Theorem 4.1**: If $L \in CDPL$ (Cooperating Distributed Pattern Language) is an infinite language over $\sum$, then there is $u € \sum^+$ such that for all $n \geq 1$a string of the form $w_1 u^{kn} w_2$ is in $\sum^+$, where $w_1, w_2 \in \sum^+$ and k is the number of discrete derivation steps in succession in the grammar system.

**Proof:** Consider a CDPG system $\Gamma = (\Sigma, G_1, G_2, \ldots, G_m, X, A)$. Let us take a component grammar $G_i = (\sum_i, X_i, P_i)$. As L is infinite the length of the pattern must be greater than 1 (i.e) $|p_i| > 1$. For such pattern $p_i$ we distinguish several cases:

(i) $p_i = u\delta x$, $u \in \sum_i^+$, $\delta \in X_i$, $x \in (\sum_I \cup X_i)^*$. Clearly, L(G) contains al strings of the form $u^k z y_n$, $k \geq 1$, $z \in A$, $y_k$ is obtained by replacing $\delta$ in $p_i$ by z, each variable, if any, in x by strings in A, then replacing $\delta$ in $p_i$ by the result and each variable in x, if any, by strings in A and repeating this operation n times.
(ii) $p_i = = x\delta u$, $u \in \sum_i^+$, $\delta \in X_i$, $x \in (\sum_i \cup X_i)^*$. The reversed situation is obtained.
(iii) $p_i = = \delta_{1i} x \delta_{2i}$, $\delta_{1i}, \delta_{2i} \in X_i$ equal or not, $x \in (\sum_I \cup X_i)^+$. Let $z_1$, $z_2$ be in A and y be obtained by replacing in x all variables, if any, by strings in A.

Then all strings of the form $u^k v$ or $vu^k$ is generated in the component grammar $G_i$. Now as the CDP grammar system $\Gamma = (\Sigma, G_1, G_2, \ldots, G_m, X)$ consists of m components, the strings generated in the component $G_i$ are used as axioms by some other component, and these strings are used by yet another component. This process is repeated n number of times. Thus at the end, the language generated by the system $\Gamma = (\Sigma, G_1, G_2, \ldots, G_n, X, A)$ consists of words $w_1 u^{kn} w_2$.

## 5. Learning Cooperating Distributed Pattern Grammar System

Consider the situation where the learning algorithm is allowed to make queries to an oracle. In [4], the notion of "minimally adequate teacher" (MAT) is introduced and the teacher (Oracle) answers membership and equivalence queries in order to construct a learning algorithm for regular sets. In [5], the notions of subset and superset queries are introduced. For a subset (superset) query, the input is a concept C and the output is 'yes' if C is a subset (superset) of the target concept $C_*$ and 'no' otherwise. If the answer is 'no', counter example x from $C - C_*$ ($C_* - C$) is also returned. Restricted subset queries and restricted superset queries, where no counter example is returned are also introduced in [5].

We try to learn a CDPGS grammar system with a single pattern which is in **canonical form**. The technique of the algorithm is as follows: First, the pattern of the pattern grammar is learnt using prefix queries and the axioms are learnt using restricted subset queries.

We recall that a word $u \in \Sigma^*$ is a prefix of another word $w \in \Sigma^*$, if there exists a word $v \in \Sigma^*$, such that $w = uv$. Thus in a prefix query, the concept to be learnt is usually a word w over the underlying alphabet T. The input is a word $u \in \Sigma^*$ and the output is "yes", if u is a prefix of w and "no" otherwise. The class of all k variable patterns is denoted by $P_k$.

We now present an algorithm that exactly identifies in polynomial time the class $P_k$.of pattern languages using prefix queries. Let $p = p_1p_2\ldots p_n$ be the pattern to be identified. We begin by checking whether $p_1$ is a constant. Hence for each $a \in \Sigma$ we make a prefix query for a.. If the output is "no" to each of these queries we conclude that $p_1$ is a variable and since p is in canonical form $p_1 = x_1$.

Suppose at some stage we have discovered that $p_1p_2\ p_3\ldots p_i$ is a prefix of p and $j = \max\ \{r\ /\ p_s = x_r \text{ for } 1 \le s \le i\}$. Again we check whether $p_{i+1}$ is a constant by making prefix query for $p_1\ p_2\ p_3\ldots\ p_i\, a,\ a \in \Sigma$. As before if each of these queries yields a negative answer, we conclude that $p_{i+1}$ is a variable and query whether $p_1\ p_2\ldots\ p_i\, x_r$ is a prefix for each $r \le j + 1$. We conclude that the pattern is complete if each of these queries receives a negative reply. Now, to learn axiom set A, initially fix $A = \phi$. Arrange the words in $\bigcup_{i=1}^{m}\Sigma^i$ (m the maximum length of the axiom is known) are arranged according to increasing order of length and among the words of equal length lexicographically. Let them be $x_1, x_2\ldots\ x_s$. At the t^th step ask the restricted subset query for $(T, A \cup \{x_t\}, p)$. If the answer is 'yes', increment A to $A \cup \{x_t\}$. If the answer is 'no', A is not incremented. The output at the last step is the required PG.

The advantage of this learning is, a sample word from the language generated by the system is not needed to learn the system as is done in parallel communication.

**Algorithm**
**Input:**
The alphabets $\Sigma_j$, $X_j$, a positive sample $w \in \Sigma_j^+$ of length r with $w = w_1w_2\ldots.w_r$, the length 'n' of the pattern, the maximum length 'm' of the axiom, $r \ge n$ , words $t_1, t_2\ldots, t_s$ of $\bigcup_{i=10}^{m}\Sigma_j^{\,i}$ given in the increasing length order, among words of equal length according to lexicographic order.

**Output:**
A cooperating distributed pattern grammar system $\Gamma' = (\Sigma, G_1, G_2, X, A)$ with $L(\Gamma') = L(\Gamma)$

**Procedure (Pattern 1)**

```
Begin
J = 1

Module 1
 i = 0, p = λ, number of characters in the pattern is n
First set
        for a Є $\Sigma_j$
        begin
                Ask prefix query for pa
                  if answer is "yes" then
                    begin
                         p = pa
                           i = i +1
                             if i is equal to n
                           begin
                                  exit
                           end
                           call first set
                              else
                                   call module 2
                         end
                end
Module 2
Second set
        for x Є $X_j$
        begin
                Ask prefix query for px
                  if answer is "yes" then
                    begin
                         p = px
                           i = i + 1
                             if I is equal to n
                begin
                         exit
                end
                call second set
        else
                call module 1
        end
        end
      end
```

**Procedure (Axiom)**
Let $x_1,\ x_2,\ \ldots,\ x_s$ be the words in $\bigcup_{i=1}^{m}\Sigma_j^{\,i}$ arranged in lexicographic order

```
A = ϕ
        for t = 1 to s do
                begin
                         ask restricted subset query for
G = ($\Sigma_j$ , A ∪ {$x_t$}, {p})
```

```
                    If 'yes' then $A = A \cup \{x\}$ and $t = t + 1$
                else output G
            end
Print the Cooperating grammar ($\Sigma_j$ ,$X_j$, $p_j$)
```

**Procedure (Pattern 2)**

```
Begin
J = j +1

Module 1
 i = 0, p = $\lambda$, number of characters in the pattern is n
First set
    for a Є $\Sigma_j$
    begin
        Ask prefix query for pa
          if answer is "yes" then
            begin
                p = pa
                  i = i +1
                    if i is equal to n
                    begin
                        exit
                    end
                    call first set
                      else
                        call module 2
                end
        end
Module 2
Second set
    for x Є $X_j$
    begin
        Ask prefix query for px
          if answer is "yes" then
            begin
                p = px
                  i = i + 1
                    if I is equal to n
        begin
                exit
        end
        call second set
    else
        call module 1
      end
      end
    end
```

## 6. Example run for PC (PL)

Let us consider a CDPG system given in the example 4.2 with two components, which are pattern grammars. First we try to learn pattern grammar 1. To learn a pattern grammar it is enough if we find the pattern p and the axiom set A. Here the length of the pattern 1 is two and maximum length of the axiom is one and the alphabet $\Sigma = \{a, b, c\}$. Let $p = p_1p_2$. First we check whether $p_1$ is a constant. Thus for $a \in \Sigma_1$, a prefix query is asked. The answer will be "no" since the pattern is $\delta_1 b$. Thus $p_1 = \delta_1$ is learnt. Now for $a \in \Sigma_1$ we ask a prefix query for $\delta_1 a$. The answer will be "no". Again for $b \in \Sigma_1$, a prefix query is asked for $\delta_1 b$. The answer will be "yes". Since we get answer "yes" we conclude that $p_1$ as $\delta_1 b$.

Now, to learn axiom set A, initially fix $A = \phi$. The words in $\bigcup_{i=1}^{3} \Sigma^i$ are arranged according to increasing order of length and among the words of equal length lexicographically. Let them be a, b, c, aa, bb, cc, ab, ba, ac, ca, bc, cb, abc, bca, … Now the restricted subset query for $(\Sigma, A \cup \{a\}, p_1)$ is asked. As the answer is 'no' one more subset query $(\Sigma, A \cup \{b\}, p_1)$ is asked. Here the teacher answers no. Then we ask subset query $(\Sigma, A \cup \{c\}, p_1)$. Here the teacher answers yes. Thus the axiom set is learnt which is {c}. Thus the pattern 1 and axiom are learnt.

Now since the CDPGS has two components which are pattern grammars, we try to find the second component whose pattern (pattern 2) is learnt as explained above. The axiom 2 need not be learnt because the output of the first component is the axiom of the second component.

## References

[1] D. Angluin, Finding patterns common to a set of strings, Journal of Computer and System Sciences 21(1980), 46–62.

[2] J. Dassow, Gh. Paun and G. Rozenberg, Generating languages in a distributed way: Grammar systems, in Handbook of Formal Languages (G. Rozenberg, A. Salomaa eds.) Springer-Verlag, Berlin, Heidelberg,, 1997

[3] J. Dassow, Gh. Paun and A. Salomaa, Grammars based on patterns, International Journal of Foundations of Computer Science 4 (1993), 1–14.

[4] S. Demitrescu and G. Paun, On the power of Parallel communicating grammar systems with right-linear components, Theoretical Informatics and Applications, 31(4), 1997, 331–354.

[5] Erzsebet Csuhaj – Varju, J. Dassow, J. Kelemen and Gh. Paun, Grammar Systems: A grammatical approach to Distribution and Cooperation, Gordon and Breach Science Publishers S. A, Switzerland, 1994.

[6] J. Gruska, The descriptional complexity of context – free languages, in Proc. MFCS Symp., High Tatras, 1973, 71–84.

[7] H. A. Maurer, A. Salomaa and D. Wood, Pure grammars, Information and Control 44 (1980), 47–72.

[8] Gh. Paun, G. Rozenberg and A. Salomaa, Pattern Grammars, Journal of Automata, Languages and Combinatorics, 1, (1996), 219–235.

[9] Gh. Paun and L. Santean, Parallel communicating grammar systems : the regular case, Ann. Univ. Buc., Serie Materm -Inform., 38(1989), 55–63.

[10] G. Rozenberg and A. Salomaa, The Mathematical Theory of L-Systerns, Academic Press, New York 1980.

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

[12] Victor Mitrana, Patterns and languages: An Overview, Grammars 2 (1999), 149 – 173.