CHAPTER 5

Pattern Grammar and Cooperating Distributed Grammar Systems

5.1 Introduction

Motivated by pattern grammars of Dassow et al. [16] and cooperating distributed grammar systems by Csuhaj-Varju et al. [12], a new grammar system, called cooperating distributed pattern grammar system (CDPGS) is introduced. In this system all the components considered are pattern grammars. The resultant family of languages is compared with other families of languages and a learning algorithm for cooperating distributed pattern grammar system is considered.

5.2 Cooperating Distributed Grammar System

The motivation of defining cooperating distributed grammar system comes from the blackboard systems. Each component grammar corresponds to a particular knowledge source of the blackboard system. The global database of the blackboard system - the blackboard - is modelled by a common sentential form in which the component grammars of the grammar system make their rewritings. The following definitions are recalled from [12] for complete understanding.
Definition 5.2.1  A cooperating distributed grammar system (CD grammar system in 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 nonterminals, the set \(T_i\) of terminals, the set \(P_i\) of context-free rules, and without axiom,

\[ T \text{ is a subset of } \bigcup_{i=1}^{n} T_i, \]

\[ S \in \bigcup_{i=1}^{n} N_i = N \]

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 that are accepted as solutions / part of solutions.

Definition 5.2.2  Let \(\Gamma\) be a CD grammar system as in Definition 5.2.1. Let \(x, y \in V_i^*\). Then we write \(x \Rightarrow^k_{G_i} y\) if and only if there are words \(x_1, x_2, \ldots, x_{k+1}\) such that

(i) \(x = x_1, y = x_{k+1}\),

(ii) \(x_j \Rightarrow_{G_i} x_{j+1}\), ie., \(x_j = x'_j A_j x''_j, x_{j+1} = x'_j w_j x''_j, A_j \rightarrow w_j \in P_i, 1 \leq j \leq k\)
Moreover, we write

\[ x \Rightarrow_{G_i}^{\leq k} y \text{ iff } x \Rightarrow_{G_i}^{k'} y \text{ for some } k' \leq k, \]

\[ x \Rightarrow_{G_i}^{\geq k} y \text{ iff } x \Rightarrow_{G_i}^{k'} y \text{ for some } k' \geq k, \]

\[ x \Rightarrow_{G_i}^{\ast} y \text{ iff } x \Rightarrow_{G_i}^{k} y \text{ for some } k, \]

\[ x \Rightarrow_{G_i}^{t} y \text{ iff } x \Rightarrow_{G_i}^{\ast} y \text{ and there is no } z \neq y \text{ with } y \Rightarrow_{G_i}^{\ast} z. \]

Any derivation \( x \Rightarrow_{G_i}^{k} y \) corresponds to \( k \) direct 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 atmost \( k \) changes. The \( \geq k \)-derivation mode represents competence since it requires that the agent can perform atleast \( k \) changes, that is, she/he/it must contribute atleast \( k \) times in succession to the solving. The \( \ast \)-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 5.2.3** Let

\[ f \in \{ t, \ast, 1, 2, \ldots, \leq 1, \leq 2, \ldots, \geq 1, \geq 2, \ldots \} \]

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

\[ S = w_0 \Rightarrow_{G_{i_1}}^{f} w_1 \Rightarrow_{G_{i_2}}^{f} w_2 \Rightarrow_{G_{i_3}}^{f} \cdots \Rightarrow_{G_{i_r}}^{f} w_r = z. \]
Some examples are given in order to illustrate the concepts.

**Example 5.2.1** Let us consider the CD grammar system

$$\Gamma = ((a, b, c), G_1, G_2, S),$$

with

$$G_1 = (\{A, B\}, \{A', B', a, b, c\}, \{A \to aA'b, B \to cB', A \to ab, B \to c\}),$$

$$G_2 = (\{S, S', A', B'\}, \{A, B\}, \{S \to S', S' \to AB, A' \to A, B' \to B\})$$

Then we obtain

$$L_1(\Gamma) = L_\ast(\Gamma) = L_{\leq k}(\Gamma) = L_{\geq 1}(\Gamma) = \{a^nb^m|n \geq 1, m \geq 1\}, k \geq 1,$$

$$L_2(\Gamma) = L_{\geq 2}(\Gamma) = \{a^nb^m|n \geq 1, n \geq 1\},$$

$$L_k(\Gamma) = L_{\geq k}(\Gamma) = \phi \text{ for } k \geq 3.$$

We show the relation only for $L_t(\Gamma)$ and $L_2(\Gamma)$; the proofs for the other relations are analogous.

It is clear that the derivation starts with $S \Rightarrow S' \Rightarrow AB$ 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. $AB \Rightarrow^2 abc$.
- Case 1. $AB \Rightarrow^2 abcB'$.
- Case 2. $AB \Rightarrow^2 aA'bc$.
- Case 3. $AB \Rightarrow^2 aA'bcB'$.

In the Case 0 the derivation has been terminated.
First we consider the derivation in the $t$-mode.

In the Case 1 we can continue only in the following way:

$$
abcB' \Rightarrow^t abcB \Rightarrow^t abccB' \Rightarrow^t abc^3B' \Rightarrow^t abc\cdots \Rightarrow^t abc^{r-2}B
$$

$$
\Rightarrow^t abc^{r-1}B' \Rightarrow^t abc^{r-1}B \Rightarrow^t abc^r.
$$

Analogously, in the Case 2 only words and all words of the form $a^rb^rc$, $r \geq 2$, can be generated.

In the 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

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

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

$$S \Rightarrow^2 AB \Rightarrow^2 aA'bcB' \Rightarrow^2 aAbcB \Rightarrow^2 a^2A'b^2c^2B' \Rightarrow^2 a^2Ab^2c^2B \Rightarrow^2 \cdots \Rightarrow^2$$

$$
\Rightarrow^2 a^{n-1}b^{n-1}c^{n-1}B \Rightarrow^2 abc^{r-1}B' \Rightarrow^t a^n b^n c^n.
$$

This proves

$$L_2(\Gamma) = \{a^n b^n c^n / n \geq 1\}.$$
5.3 Cooperating Distributed Pattern Grammar System

A new model of cooperating distributed grammar system which has \( n \) pattern grammars as components is defined \[47\].

**Definition 5.3.1** A cooperating distributed pattern grammar system with \( n \) components (CD\(_n\)PGS in 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 \( G_i = (\Sigma_i, X_i, P_i) \) is a pattern grammar where \( \Sigma_i \subseteq \Sigma \) is an alphabet whose elements are called constants, \( X_i \subseteq X \) is an alphabet whose elements are called variables. \( A \subseteq \Sigma^* \) 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 at least one variable. The grammars \( G_i \), \( 1 \leq i \leq n \), are called the components of \( \Gamma \). The rewriting in the \( i^{th} \) component is done as follows: If \( L \) is the set of words given as input to the \( i^{th} \) component then, in a single step, the language \( P_i(L) \) is got from \( P_i \), where

\[
P_i(L) = \left\{ u_1x_{i_1}u_2x_{i_2} \ldots u_kx_{i_k}u_{k+1}/u_1\delta_{i_1}u_2\delta_{i_2} \ldots u_k\delta_{i_k}u_{k+1} \in P_i, u_i \in \Sigma^*, 1 \leq i \leq k + 1, x_{i_k} \in L, \delta_{i_k} \in X_i, 1 \leq i \leq k \right\}
\]

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 \( i^{th} \) component works for \( j \) steps continuously, then we write \( y \in P_i^j(x) \), if there are words \( x_1, x_2, \ldots, x_{j+1} \) such that

(i) \( x = x_1, y = x_{j+1} \),

(ii) \( x_{r+1} \in P_i(\{x_r\}), r = 1, 2, \ldots, k \)

Moreover, we write
\( y \in P^*_i(x) \iff y \in P^j_i(x) \) for some \( j \).

The *-mode is the case where the agent can work at the blackboard as long as she/he/it wants to do.

In CD\(_n\)PGS we start a derivation from the first component and proceed with the next component and so on. The set of words derived in the last component is given as axiom set to the first component. Words that are collected in the last component form the required language. Also \( j \) represents the number of rewriting steps each component of the system does in which case \( \Gamma \) is working in \( j \)-mode. Few examples for CD\(_2\)PGS are now given.

**Example 5.3.1** Let \( \Gamma = (\{a, b, c\}, G_1, G_2, \{\delta_1, \delta_2\}, \{c\}) \), where

\[
G_1 = (\{a\}, \{\delta_1\}, \{a\delta_1\}),
\]

\[
G_2 = (\{b\}, \{\delta_2\}, \{\delta_2b\}).
\]

If each component does one rewriting, that is, \( j = 1 \), we have \( L(\Gamma) = \{a^n cb^n / n \geq 1\} \).

The derivation is as follows. At the beginning the first component produces a word ‘ac’, 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 and thus the word ‘\( a^2 cb \)’ is produced, which is treated as axiom for the next derivation in the second component and the word \( a^2 cb^2 \) is generated. The words derived at the second component form the language \( \Gamma \). After \( n \) pairs of derivations the word generated in the second component is \( a^n cb^n \).

Similarly if \( j = 2 \), the language generated is \( \{a^{2n} cb^{2n} / n \geq 1\} \). In general, if
there are $j$ direct derivation steps in succession in the grammar $G_i$, the language generated is $\{a^{jn}cb^{jn}/n \geq 1\}$.

**Example 5.3.2** Let $\Gamma = (\{a, b\}, G_1, G_2, \{\delta_1, \delta_2\}, \{\lambda\})$, where

$G_1 = (\{a\}, \{\delta_1\}, \{a\delta_1\})$

$G_2 = (\{b\}, \{\delta_2\}, \{\delta_2b\})$.

If $j = 1$, that is, each component does one rewriting, we have $L(\Gamma) = \{a^n b^n/n \geq 1\}$.

**Example 5.3.3** Let $\Gamma = (\{a, b, c\}, G_1, G_2, \{\delta_1, \delta_2\}, \{c\})$, where

$G_1 = (\{a, b, c\}, \{\delta_1\}, \{a\delta_1 a, b\delta_1 b\})$,

$G_2 = (\{a, b, c\}, \{\delta_2\}, \{a\delta_2 a, b\delta_2 b\})$.

$L(\Gamma) = \{wcw^R/w \in \{a, b\}^+\}$.

The derivation is as follows: At the beginning the first component produces words ‘aca’ and ‘bcb’, and then these words are taken as the axiom set for the second component and the words ‘$a^2ca^2$’, ‘bacab’, ‘abcba’ and ‘$b^2cb^2$’ are produced. Now this set of words is taken as axiom set for the first component and thus the words ‘$a^3ca^3$’, ‘abacaba’, ‘$a^2bcba^2$’, ‘$ab^2cb^2a$’, ‘$ba^2ca^2b$’, ‘$b^2acab^2$’, ‘$babcbab$’ and ‘$b^3cb^3$’ are generated. The language generated is $\{wcw^R/w \in \{a, b\}^+\}$.

**Example 5.3.4** Let $\Gamma = (\{a, b\}, G_1, G_2, \{\delta_1, \delta_2\}, \{b\})$, where

$G_1 = (\{a\}, \{\delta_1\}, \{a\delta_1\})$,

$G_2 = (\{b\}, \{\delta\}, \{\delta_2b\})$.

$L(\Gamma) = \{a^n b^m/n, m \geq 1\}$.

If there is no restriction on the number of steps, then the language generated is $\{a^n b^m/n, m \geq 1\}$. 101
A few examples for CD₃PG systems are now considered.

**Example 5.3.5** A language generated by a CD₂PG system may be generated by a CD₃PG system.

Let \( \Gamma = (\{a, b\}, G_1, G_2, G_3, \{\delta_1, \delta_2, \delta_3\}, \{\lambda\}) \), where

\[
G_1 = (\{a\}, \{\delta_1\}, \{a\delta_1\}),
\]

\[
G_2 = (\{b\}, \{\delta_2\}, \{\delta_2b\}),
\]

\[
G_3 = (\{a, b\}, \{\delta_3\}, \{\delta_3\}).
\]

If \( j = 1 \), that is, each component does one rewriting, then

\[
L(\Gamma) = \{a^l b^l / l \geq 1\}.
\]

If \( j = 2 \), that is, each component does two rewritings, then

\[
L(\Gamma) = \{a^{2l} b^{2l} / l \geq 1\}.
\]

If there are \( j \) rewritings in each component, then

\[
L(\Gamma) = \{a^{jl} b^{jl} / l \geq 1\}.
\]

**Example 5.3.6** Let \( \Gamma = (\{a, b\}, G_1, G_2, G_3, \{\delta_1, \delta_2, \delta_3\}, \{\lambda\}) \), where

\[
G_1 = (\{a\}, \{\delta_1\}, \{c\delta_1\}),
\]

\[
G_2 = (\{b\}, \{\delta_2\}, \{b\delta_2b\}),
\]

\[
G_3 = (\{a, b\}, \{\delta_3\}, \{a\delta_3\}).
\]

If \( j = 1 \), that is, each component does one rewriting, then

\[
L(\Gamma) = \{(abc)^l b^l / l \geq 1\}.
\]

If \( j = 2 \), that is, each component does two rewritings, then

\[
L(\Gamma) = \{(a^2 b^2 c^2)^l (b^2)^l / l \geq 1\}.
\]

If there are \( j \) rewritings in each component, then
\[ L(\Gamma) = \{(a^j b^j c^j)^l (b^j)^l / l \geq 1 \}. \]

The language generated by a cooperating distributed pattern grammar system is denoted by CDPL and the family of languages generated by a cooperating distributed pattern grammar system by CDPL. The family of languages generated by CD_nPG (with \( n \) components) is represented by CD_nPL, \( n = 1, 2, \ldots \).

**Proposition 5.3.1**

(i) \( CD(PL) \cap PL \neq \phi \).

(ii) \( CD(PL) \cap CSL \neq \phi \).

**Proof.**

(i) The language \( \{a^n b^n / n, m \geq 1 \} \) is in CDPL and PL.

(ii) The language \( \{a^{2^n} / n \geq 1 \} \) is in CDPL and CSL. \( \square \)

**Theorem 5.3.1** \( CD_1 PL \subset CD_2 PL \subset CD_3 PL \subset \cdots \subset CD_n PL \).

**Proof.** Any language generated by a CD_nPG system can also be generated by a CD_{n+1}PG system. This is seen from Examples 5.3.2 and 5.3.5. The language \( \{a^j b^j / l \geq 1 \} \) which is generated by a cooperating distributed grammar system with two components and with \( j \) rewriting steps in each component is also generated by a cooperating distributed grammar system with three components and with \( j \) rewriting steps in each component. The extra component does not produce any new word. In a similar manner, any language generated by a CD_nPG system can also be generated by a CD_{n+1}PG system, by adding an extra component which does not produce any new word. That is, \( CD_n PL \subseteq CD_{n+1} PL \).
The inclusion is a strict inclusion. This is seen from Example 5.3.6. The language \(\{(a^2b^2c^2)^l(b^2)^l/l \geq 1\}\) is generated by a cooperating distributed grammar system with three components and two rewriting steps in each component. But this language \(\{(a^2b^2c^2)^l(b^2)^l/l \geq 1\}\) cannot be generated by a cooperating distributed grammar system with two components and two rewriting steps in each component. If more patterns are considered in the first and second components, then words which are not in the form \((a^2b^2c^2)^l(b^2)^l\) are got. Therefore \(\text{CD}_nPL \subset \text{CD}_{n+1}PL\). □

**Theorem 5.3.2** If \(L \in \text{CD}_mPL\) is an infinite language \(L \subseteq \Sigma^*\), then there are \(m\) words \(u_1, u_2, \ldots, u_m \in \Sigma^+\) such that for all \(j \geq 1\), a string of the form \(u_m^ju_{m-1}^j\ldots u_2^jv_m\) or \(v_mu_1^jv_2^j\ldots u_m^j\) is in \(L\).

**Proof.** Consider a CD\(_m\)PG system \(\Gamma = (\Sigma, G_1, G_2, \ldots, G_m, X, A)\). Let us take the first component grammar \(G_1 = (\Sigma_1, X_1, P_1)\), as in a CDPG system the derivation starts from the first component. Any pattern in \(P_1\) of the first component \(G_1\) is distinguished in several cases:

(i) \(p = u_1\delta_1x_1, \ u_1 \in \Sigma_1^+, \ \delta_1 \in X_1, \ x_1 \in (\Sigma_1 \cup X_1)^*\). Clearly, \(L(G_1)\) contains all strings of the form \(u_1^jzy_j, \ j \geq 1, \ z \in A, \ y_j\) is obtained by replacing \(\delta_1\) in \(p\) by \(z\), each variable, if any, in \(x_1\) by strings in \(A\), then replacing \(\delta_1\) in \(p\) by the result and each variable in \(x_1\), if any, by strings in \(A\) and repeating this operation \(j\) times.

(ii) \(p = x_1\delta_1u_1, \ u_1 \in \Sigma_1^+, \ \delta_1 \in X_1, \ x_1 \in (\Sigma_1 \cup X_1)^*\). The reversed situation is obtained.

(iii) \(p = \delta_{11}x_1\delta_{21}, \ \delta_{11}, \delta_{21} \in X_1\) equal or not, \(x_1 \in (\Sigma_1 \cup X_1)^+\). Let \(z_1, z_2\) be in

104
Then all strings of the form \((z_1y)^jz_2, \ j \geq 1\) are in \(L(G_1)\). Taking \(u_1 = z_1y, v_1 = z_2\) we have \(u_1^jv_1 \in L(G_1), \ j \geq 1\).

Then all strings of the form \(u_1^jv_1\) or \(v_1u_1^j\) are generated in the component grammar \(G_1\). Now as the CD\(_m\)PG grammar system \(\Gamma = (\Sigma, G_1, G_2, \ldots, G_m, X)\) consists of \(m\) components, the strings generated in the component \(G_1\) are used as axioms by the second component \(G_2\), and any pattern in \(P_2\) can be distinguished in several manner.

Hence words of the form \(u_2^jv_2\) or \(v_2u_2^j\) are generated in the second component. Proceeding in this manner \(u_1^j u_2^j \ldots u_m^j v_m\) or \(v_m u_1^j u_2^j \ldots u_m^j\) are generated in the \(m^{th}\) component. Thus a string of the form \(u_1^j u_2^j \ldots u_m^j v_m\) or \(v_m u_1^j u_2^j \ldots u_m^j\) is in \(L\).

5.4 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 learn a CDPG system with a single pattern in each component. 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 \( P_k \).

An algorithm is presented now that exactly identifies the class \( P_k \) 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 \( p_1p_2p_3 \ldots p_i \) is a prefix of \( p \) and \( j = \max\{r/p_s = x_r \text{ for } 1 \leq s \leq i \} \). Again it is checked whether \( p_{i+1} \) is a constant by making prefix query for \( p_1p_2p_3 \ldots p_i a, a \in \Sigma \). As before if each of these queries yields a negative answer, we conclude that \( p_{i+1} \) is a variable and query whether \( p_1p_2 \ldots p_ix_r \) is a prefix for each \( r \leq j + 1 \). It is concluded 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 length of the longest axiom is known) 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 pattern grammar.

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

\[
w = w_1w_2 \ldots w_r, \text{ the length ‘}n\text{’ of the pattern, the maximum length ‘}m\text{’ of the axiom, } r \geq n, \text{ words } t_1, t_2, \ldots, t_s \text{ of } \bigcup_{i=1}^{m} \Sigma_j^i \text{ given in the increasing length order, among words of equal length according to lexicographic order.}
\]

**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**

\( i = 0, \ p = \lambda, \) number of characters in the pattern is \( n \)

First set

for \( a \in \Sigma_j \)

begin
Ask prefix query for $pa$
if the answer is “yes” then
begin
\[ p = pa \]
\[ i = i + 1 \]
if $i$ is equal to $n$
begin
\hspace{1cm} \text{exit} \]
end
\hspace{1cm} \text{call first set} \]
else
\hspace{1cm} \text{call module 2} \]
end
end

Module 2
Second set
for $x \in X_j$
begin
Ask prefix query for $px$
if the answer is “yes” then
begin
\[ p = px \]
end
\[ i = i + 1 \]

if \( i \) is equal to \( n \)
begin
exit
end

call second set
else
call module 1
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

\[ A = \emptyset \]

for \( t = 1 \) to \( s \) do
begin
ask restricted subset query for \( G = (S_j, A \cup \{x_t\}, \{p\}) \)
If ‘yes’ then \( A = A \cup \{x_t\} \) and \( t = t + 1 \)
else output \( G \)
end
Print the cooperating grammar \( (S_j, X_j, p_j) \)
Procedure (Pattern 2)

begin

\[ j = j + 1 \]

Module 1

\[ i = 0, p = \lambda, \text{number of characters in the pattern is } n \]

First set

for \[ a \in \Sigma_j \]

begin

Ask prefix query for \( pa \)

if the 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

end
Module 2

Second set

for $x \in X_j$

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
5.5 Example Run

A CDPG system given in the Example 5.3.1 with two components, which are pattern grammars are considered now. 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 of the first component is two and maximum length of the axiom is one and the alphabet is $\Sigma = \{a, b, c\}$. Let $p = p_1p_2$. First it is checked whether $p_1$ is a constant. Thus for $a \in \Sigma_1$, a prefix query is asked. The answer is “yes” since the pattern is $a\delta_1$. Thus $p_1 = a$ is learnt.

Now for $a \in \Sigma_1$ again a prefix query for $aa$ is asked. The answer is “no”. This shows that $p_2 = \delta_1$. Thus the pattern $p = a\delta_1$ is learnt.

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 $a, b, c, aa, bb, cc, ab, ba, ac, ca, bc, cb, abc, bca, \ldots$. 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 the subset query $(\Sigma, A \cup \{c\}, p_1)$ is asked. Here the teacher answers yes. Thus the axiom set is learnt which is $\{c\}$. Thus the pattern of the first component and axiom of the grammar system are learnt. Now since the CDPGS has two components which are pattern grammars, the second component is learnt as explained above. The axiom of the second component need not be learnt because the output of the first component is the axiom of the second component. Now as the patterns of the two components and the axiom of the grammar system are known, the output is a cooperating distributed pattern grammar system $\Gamma = (\Sigma, G_1, G_2, X, A)$. 