CHAPTER 2

Synchronized Pure Pattern Grammar

and Parallel Communication

2.1 Introduction

In formal language theory, studies related to language generation use techniques that are based on notions in DNA computing. One such notion is that of a filter.

Parallel communicating grammar systems with communication by command (CCPC grammar systems, in short) [13] represent the first model of networks of language processors where communication is performed through filters. A CCPC grammar system consists of a finite set of Chomsky grammars defined over a common nonterminal alphabet and a common terminal alphabet. The system with filter or selector language functions by alternating rewriting and communication steps. The grammars start working from their own start symbols. A rewriting step in these systems is defined as follows. Each grammar generates its own string until it has no more applicable productions. Then the components communicate their strings to each other in the following manner. Every grammar tries to send a copy of its string to each of the other grammars, but only those strings are accepted at a component which pass through the filter associated with that component.
Motivated by the study of pure pattern grammars by Abisha et al. [2], parallel communicating grammar systems by Păun et al. [40] and parallel communicating grammar systems with communication by commands of Csuhaj-Varju et al. [13], a new generative device, referred as parallel communicating synchronized pure pattern grammar system with filters is introduced [46]. In this grammar system, all components except the master are synchronized pure pattern grammars, but the master is a regular grammar. This new generative device is compared with the earlier models [2, 13] and found to have different properties.

2.2 Preliminaries

In this section, parallel communicating grammar systems (PC grammar systems) using synchronized pure pattern grammars are considered. The PC grammar system is recalled for completeness.

**Definition 2.2.1** [40] A centralized PC grammar system of degree \( n, n \geq 1 \) is a construct

\[
\Gamma = (N, T, K, (P_1, S_1), (P_2, S_2), \ldots, (P_n, S_n))
\]

where \( N, T, K \) are pairwise disjoint alphabets, with \( K = \{Q_1, Q_2, \ldots, Q_k\} \), \( S_i \in N \) and \( P_i \) are finite sets of rewriting rules over \( N \cup T \cup K \), \( 1 \leq i \leq n \); the elements of \( N \) are nonterminal symbols, the elements of \( T \) are terminals; the elements of \( K \) are called query symbols; the pairs \((P_i, S_i)\) are the components of the system. It is observed that, by their indices, the query symbols are associated with the components.
For \((x_1, \ldots, x_n), (y_1, \ldots, y_n)\) with \(x_i, y_i \in (N \cup T \cup K)^*, 1 \leq i \leq n\), (such an \(n\)-tuple is called as a configuration), and is written as \((x_1, \ldots, x_n) \Rightarrow_r (y_1, \ldots, y_n)\) if one of the following two cases holds:

1. If there is no query symbol in \(x_i\), \(x_i \Rightarrow_P y_i\) or \(x_i = y_i \in T^*, 1 \leq i \leq n\)

2. If \(x_i\) has a query symbol, that is \(x_i = z_1Q_{i1}z_2Q_{i2} \ldots z_tQ_{it}z_{t+1}\) for \(t \geq 1, z_k \in (N \cup T)^*, 1 \leq k \leq t + 1\), then \(y_i = z_1x_{i1}z_2x_{i2} \ldots z_tx_{it}z_{t+1}\) [and \(y_{ij} = S_{ij}\), \(1 \leq j \leq t\)], where \(x_{ij}\) is the string of \(i_j^{th}\) component \(j = 1, 2, \ldots, t\). For all \(i_j^{th}\) component with unspecified \(i\), we have \(y_i = x_i\).

Condition (1) defines a rewriting step (component wise, synchronously, using one rule in every component whose current string is not only on terminals), and condition (2) defines a communication step; the query symbols \(Q_{ij}\) introduced in \(x_i\) are replaced by the associated strings \(x_{ij}\). The communication has priority over rewriting. The work of the system is blocked when no query symbol is present but condition (1) is not realized because a component cannot rewrite its sentential form, although there is a nonterminal in the string.

The above relation \(\Rightarrow_r\) is said to be performed in the returning mode; after communication, a component resumes working from its axioms. If the brackets [and \(y_{ij} = S_{ij}\), \(1 \leq i \leq t\)], are removed, then the non-returning mode of derivation is obtained. After communication, a component continues the processing of the current string. The obtained relation is denoted by \(\Rightarrow_{nr}\). The language generated by \(\Gamma\) is the language generated by its first component when starting from \((S_1, \ldots, S_n)\), that is, \(L_f(\Gamma) = \{w \in T^*/(S_1, \ldots, S_n) \Rightarrow_r (w, \alpha_2, \ldots, \alpha_n), \text{for } \alpha_i \in (N \cup T)^*, 2 \leq i \leq n\}\).
\( f \in \{ r, nr \} \). The first component is called the master component of the system.

The above model is called centralized if one component (1st component) acts as the master and asks for the results of the other components by making queries.

### 2.3 Parallel Communicating Synchronized Pure Pattern Grammar Systems

A centralized parallel communicating grammar system is now considered where the first component (the master component) of the system is either a regular or a context-free grammar and the other components are synchronized pure pattern grammars.

**Definition 2.3.1** A PC (SPPG) system is a construct

\[
\Gamma = (N, T, K, (P_0, S_0), (A_1, P_1), \ldots, (A_n, P_n)),
\]

where \( N, T, K \) are pairwise disjoint alphabets with \( K = \{ Q_1, Q_2, \ldots, Q_n \} \), \( S_0 \in N \), \((N \cup K, T, P_0, S_0)\) is a regular grammar and \((T, A_i, P_i)\) are SPPG, \( i = 1 \) to \( n \).

The rewriting is similar to that of PC grammar systems with the following modifications.

Initial configuration is \((S_0, w_1, w_2, \ldots, w_n)\) where \( w_i \in A_i \), the axiom set of the \( i^{th} \) component. The rewriting in the component \((T, A_i, P_i)\) is done according to the SPPG, that is, any word from \( P^t_i(A_i) \) is considered in the \( l^{th} \) step, until a query is asked. If a query symbol \( Q_j \) appears in the master component, then the string in the \( j^{th} \) component is communicated. After communication, the components resume working from their axioms if in returning mode ‘\( r \)’ or the components continue the
processing of the current strings if they are in non-returning mode ‘nr’.

**Example 2.3.1** Now an example of PC(SPPG) is given.

\[ \Gamma = (N, T, K, (P_0, S_0), (A_1, P_1)) \]

where

\[ N = \{S_0, A\}; \]

\[ T = \{a, b\}; \]

\[ K = \{Q_1\}; \]

\[ P_0 = \{S_0 \rightarrow bA, A \rightarrow aA, A \rightarrow aQ_1\}; \]

\[ P_1 = \{aa\}, \]

\[ A_1 = \{a\}, \]

\[ (S_0, a) \Rightarrow_f (bA, a^2) \Rightarrow_f (baA, a^4) \Rightarrow_f \cdots \Rightarrow_f^* (ba^nQ_1, a^{2n+1}) \Rightarrow_f (ba^n a^{2n+1}, y), \]

\[ L(\Gamma) = \{ba^{(n+2n+1)}/n = 1, 2, \ldots\} \]

where \( y = a \) if \( f = r \); \( y = a^{2n+1} \) if \( f = nr \).

### 2.4 Parallel Communicating Grammar Systems with Communication by Commands

Parallel communicating grammar systems with communication by commands by Csuha-Varthu et al. [13], represent the first model of network of language processors where communication is performed through filters. A rewriting step in these systems is defined as follows. Each grammar generates its own string until it has no more applicable productions. Then the components communicate their strings to each other in the following manner. Every grammar sends a copy of its string to each of the other grammar, but only those strings are accepted at a component which pass
through the filter associated with that component.

**Definition 2.4.1** [13] A parallel communicating grammar system with communication by command and with finite sets of axioms or an FCCPC grammar system (of degree \(n\)) is a construct

\[
\Gamma = (N, T, (F_1, P_1, R_1), (F_2, P_2, R_2), \ldots, (F_n, P_n, R_n)), \quad n \geq 1,
\]

where \(N\) and \(T\) are disjoint finite alphabets; \(N\) is called the nonterminal alphabet and \(T\) is called the terminal alphabet of the system, \((F_i, P_i, R_i), 1 \leq i \leq n\), is the \(i^{th}\) component of \(\Gamma\), where \(F_i \subset (N \cup T)^*\) is a nonempty finite set called the set of axioms of the components, \(P_i\) is a finite set of rewriting rules over \(N \cup T\), and \(R_i \subseteq (N \cup T)^*\) is a regular language, called the selector language or the filter of the \(i^{th}\) component. The first component is designated as the master component.

**Definition 2.4.2** [13] By a configuration (a state) of an FCCPC grammar system \(\Gamma = (N, T, (F_1, P_1, R_1), \ldots, (F_n, P_n, R_n)), n \geq 1\), we mean an \(n\)-tuple \((L_1, \ldots, L_n)\), where \(L_i \subset (N \cup T)^*, 1 \leq i \leq n\), are finite languages. \((F_1, \ldots, F_n)\) is called the initial configuration (the initial state) of \(\Gamma\). FCCPC grammar systems function by changing their configurations. This is realized by alternating rewriting and communication steps.

**Definition 2.4.3** [13] Let \(\Gamma = (N, T, (F_1, P_1, R_1), (F_2, P_2, R_2), \ldots, (F_n, P_n, R_n)), n \geq 1\), be an FCCPC grammar system and let \(C_1 = (L_1, \ldots, L_n)\) and \(C_2 = (L'_1, \ldots, L'_n)\) be two configurations of \(\Gamma\). Here \(C_2\) is derived from \(C_1\) by a rewriting step, denoted by \((L_1, \ldots, L_n) \Rightarrow_{\Gamma} (L'_1, \ldots, L'_n)\), if for all \(i, 1 \leq i \leq n\).
\[ L'_i = \{ y/x \Rightarrow^*_p y, x \in L_i \text{ and there is no } z \in (N \cup T)^* \text{ such that } y \Rightarrow^*_p z \} \].

It is noted that if \( x \in L_i \cap T^* \), then \( x \in L'_i \).

Thus after a rewriting step, the new set of strings consists of all words which can be generated with zero, one or more derivation steps in the grammar from the strings of the component and the grammar has no rule applicable to any of these strings. It is clear that the synchronization of the rewriting steps is done at the level of the FCCPC grammar system; the individual components may perform a different number of rewriting steps on the different strings.

**Definition 2.4.4** [13] Let \( \Gamma = (N, T, (F_1, P_1, R_1), \ldots, (F_n, P_n, R_n)) \), when \( n \geq 1 \), be an FCCPC grammar system and let \( C_1 = (L_1, \ldots, L_n) \) and \( C_2 = (L'_1, \ldots, L'_n) \) be two configurations of \( \Gamma \). Here \( C_2 \) is derived from \( C_1 \) by a communication step, denoted by \( (L_1, \ldots, L_n) \vdash_\Gamma (L'_1, \ldots, L'_n) \) if the following conditions hold. Let the set of strings sent by the \( i^{th} \) component to the \( j^{th} \) one be defined as

\[
\delta(L_i, j) = \begin{cases} 
\{\lambda\}, & \text{if } L_i \cap R_j = \emptyset \text{ or } i = j \\
L_i \cap R_j, & \text{otherwise},
\end{cases}
\]

for \( 1 \leq i, j \leq n \). Let

\[
\Delta(j) = \delta(L_1, j)\delta(L_2, j) \ldots \delta(L_n, j),
\]

be the concatenation of the sets of strings sent to the \( j^{th} \) component, for \( 1 \leq j \leq n \). \( \Delta(j) \) is the “total message” received by the \( j^{th} \) component, and let

\[
\delta(i) = \delta(L_i, 1)\delta(L_i, 2) \ldots \delta(L_i, n),
\]

for \( 1 \leq i \leq n \), represents the “string transferring capacity” of the \( i^{th} \) component.
Then, for $1 \leq i \leq n$, it is defined

$$L'_i = \begin{cases} 
\Delta(i), & \text{if } \Delta(i) \neq \{\lambda\} \\
L_i, & \text{if } \Delta(i) = \{\lambda\} \text{ and } \delta(i) = \{\lambda\}, \\
F_i, & \text{if } \Delta(i) = \{\lambda\} \text{ and } \delta(i) \neq \{\lambda\}
\end{cases}$$

After a communication step, the obtained language, $L'_i$, is either the concatenation of the received sets of strings, or it is the previous language, when this component is not involved in communication, or it is equal to $F_i$, if this component successfully sends strings to the other components but it does not receive any string.

It is observed that a component is not allowed to send strings to itself. A sequence of alternating rewriting steps and communication steps determines a computation (a derivation) in $\Gamma$. A rewriting step and a communication step are referred as a computation (a derivation) step. The language of FCCPC grammar system is the set of terminal words which appear at the master during a computation after performing a rewriting step.

**Definition 2.4.5** [13] The language $L(\Gamma)$ generated by an FCCPC grammar system $\Gamma = (N, T, (F_1, P_1, R_1), \ldots, (F_n, P_n, R_n))$, when $n \geq 1$ is defined as follows:

$$L(\Gamma) = \left\{ w \in T^*/(F_1, \ldots, F_n) \Rightarrow_{\Gamma} (L_1^{(1)}, \ldots, L_n^{(1)}) \vdash_{\Gamma} (L_1^{(2)}, \ldots, L_n^{(2)}) \Rightarrow_{\Gamma} (L_1^{(3)}, \ldots, L_n^{(3)}) \vdash_{\Gamma} (L_1^{(4)}, \ldots, L_n^{(4)}) \Rightarrow_{\Gamma} \ldots (L_1^{(s)}, \ldots, L_n^{(s)}), \text{ for some } s \geq 1 \text{ such that } w \in L_1^{(s)} \right\}$$

38
Example 2.4.1  Consider the FCCPC grammar with two components

\[ \Gamma = (N, T, (F_1, P_1, R_1), (F_2, P_2, R_2)) \]
where

- \( N = \{A, B, C, A', B', C'', A'', B'', C'', S, S'\} \);
- \( T = \{a, b, c\} \) and
- \( F_1 = \{ABC\} \)
- \( P_1 = \{A \rightarrow aA', B \rightarrow bB', C \rightarrow cC', A'' \rightarrow \lambda, B'' \rightarrow \lambda, C'' \rightarrow \lambda\} \)
- \( R_1 = \{a^+Ab^+Bc^+C\} \cup \{a^+A''b''B''c''C''\} \)
- \( F_2 = \{S\} \)
- \( P_2 = \{S \rightarrow S', A' \rightarrow A, B' \rightarrow B, C' \rightarrow C, A' \rightarrow A'', B' \rightarrow B'', C' \rightarrow C''\} \)
- \( R_2 = a^+A'b^+B'c^+C' \)

The first few steps of the derivation are as follows:

\[(\{ABC\}, \{S\}) \Rightarrow (\{aA'bB'cC'\}, \{S'\}) \vdash (\{ABC\}, \{aA'bB'cC'\}) \Rightarrow (\{aA'bB'cC'\}, w),\]

where \( w \) is the form \( aXbYcZ \), with \( X \in \{A, A''\}, Y \in \{B, B''\}, Z \in \{C, C''\} \). This word can only be successfully communicated to the first component if it is either of the form \( w = aAbBcC' \) or \( aA''b''B''c''C'' \). In the latter case the next rewriting step results in terminal word \( abc \) at the master component, while in the first case the derivation can be continued with \( a^2A'b^2B'c^2C' \) at the first component. Analyzing the possible further derivation steps, it is easy to see \( L(\Gamma) = \{a^nb^nC^n/n \geq 1\} \) which is not a context-free language.
2.5 Parallel Communicating Synchronized Pure Pattern Grammar Systems with Filters

Now parallel communicating synchronized pure pattern grammar system with filters (CCPC(SPPG) grammar system, in short) is defined. In this new generative model, the strings at the component level are communicated to the master component only when a query symbol appears in the master component. The string is collected at the master only if it passes through a filter attached with the master component. The language is the concatenation of the received strings with strings of the master component.

**Definition 2.5.1** A CCPC(SPPG) system is a construct

$$\Gamma = (N, T, (P_0, S_0, R), (A_1, P_1), (A_2, P_2), \ldots, (A_n, P_n))$$

where $N$ and $T$ are disjoint, $S_0 \in N$, $(N, T, P_0, S_0)$ is a regular grammar, $R$ is a regular language called the selector language or the filter of the component, $Q \in N$ is a special symbol called query symbol. Each $(T, A_i, P_i)$, $i = 1, 2, 3, \ldots$ is a SPPG. The rewriting in the component $(A_i, P_i)$ is done according to the SPPG; that is, any word in $P_i^j(A_i)$ is considered at any time. If the query symbol appears in the master component then all the other components communicate their strings $P_i^j(A_i)$ to the master, if the strings can pass through the filter.

If $L_i$ is the set of strings generated in the $i^{th}$ component, then the set of strings
sent by the \( i^{th} \) component to the master component is defined as

\[
\delta(L_i) = \begin{cases} 
\{\lambda\} & \text{if } L_i \cap R = \phi, \\
L_i \cap R & \text{otherwise}
\end{cases}
\]

for \( 1 \leq i \leq n \). Let

\[
\Delta = \delta(L_1)\delta(L_2)\ldots\delta(L_n)
\]

be the concatenation of the set of strings sent to the master component, that is, the total message received by the master component.

**Definition 2.5.2** By a configuration of a CCPC(SPPG) system

\[
\Gamma = (N, T, (P_0, S_0, R), (A_1, P_1), \ldots, (A_n, P_n))
\]

we mean an \( n + 1 \) tuple \((L_0, L_1, \ldots, L_n)\) where \( L_0 \subseteq (N \cup T)^* \), \( L_i \subseteq T^* \), \( 1 \leq i \leq n \) are finite languages. \((S_0, A_1, \ldots, A_n)\) is called the initial configuration. CCPC(SPPG) grammar system functions by changing configurations. This is realized by a sequence of rewriting steps and then by a communication step.

**Definition 2.5.3** If \((L_0^{(j)}, L_1^{(j)}, \ldots, L_n^{(j)})\) is the configuration at the \( j^{th} \) step, the next configuration \((L_0^{(j+1)}, L_1^{(j+1)}, \ldots, L_n^{(j+1)})\) in the rewriting step is defined by

\[
L_0^{(j+1)} = \{\alpha \in (N \cup T)^*/ \text{there is some } \beta \in L_0^{(j)} \text{ with } \beta \Rightarrow P_0 \alpha\}
\]

and \(L_i^{(j+1)} = P_i(L_i^{(j)})\), \( i = 1, \ldots, n; \ j = 1, 2, \ldots\)

This is denoted by writing \((L_0^{(j)}, L_1^{(j)}, \ldots, L_n^{(j)}) \Rightarrow \Gamma (L_0^{(j+1)}, L_1^{(j+1)}, \ldots, L_n^{(j+1)})\).

Let \( \Rightarrow_{\Gamma}^* \) be the reflexive, transitive closure of \( \Rightarrow_{\Gamma} \).
Definition 2.5.4  If \((L_0^{(j)}, L_1^{(j)}, \ldots, L_n^{(j)})\) is the configuration at the \(j^{th}\) step and \(L_0^{(j)} \subseteq (T \cup \{Q\})^*\), then a communication step is executed. We write

\[(L_0^{(j)}, L_1^{(j)}, \ldots, L_n^{(j)}) \vdash \Gamma (L, Y_1, \ldots, Y_n)\]

Here \(L\) is obtained from \(L_0^{(j)}\) by substituting \(\Delta\) for the query symbol \(Q\) where \(\Delta = \delta(L_1^{(j)})\delta(L_2^{(j)}) \ldots \delta(L_n^{(j)})\)

\[Y_i = \begin{cases} A_i & \text{if in the returning mode} \\ L_i^{(j)} & \text{otherwise.} \end{cases}\]

Definition 2.5.5  The language \(L(\Gamma)\) generated by

\[\Gamma = (N, T, (P_0, S_0, R), (A_1, P_1), \ldots, (A_n, P_n))\] is

\[L(\Gamma) = \{w \in T^* / (S_0, A_1, \ldots, A_n) \Rightarrow^* (L_0^{(s)}, L_1^{(s)}, \ldots, L_n^{(s)}) \vdash \Gamma (L, Y_1, \ldots, Y_n) \text{ for some } s \geq 1, w \in L\}\]

Example 2.5.1

(i) \(\Gamma_1 = (N, T, (P_0, S_0, R), (A_1, P_1))\) where

\[N = \{S_0, A, Q\}\]

\(Q\) is a query symbol

\[T = \{a, b\}\]

\[P_0 = \{S_0 \rightarrow bA, A \rightarrow aA, A \rightarrow aQ\}\]

\(R\) is the language represented by the regular expression \((aa)^*\)

\[P_1 = \{ab\}\]

\[A_1 = \{a, \lambda\}\]
\[
(S_0, \{a, \lambda\}) \Rightarrow (A, \{\lambda, a, a^2\}) \Rightarrow (A, \{\lambda, a, a^2, a^3, a^4\})
\]

\[
(ba^2Q, \{\lambda, a, a^2, a^3, a^4, \ldots, a^8\}) \vdash (ba^2\{\lambda, a, a^4, a^6, a^8\}, y),
\]

\[
y = \{a, \lambda\} \text{ if } f = r; \ y = \{\lambda, a, a^2, a^4, \ldots, a^8\} \text{ if } f = nr
\]

\[
L(\Gamma_1) = \{ba^n/n \geq 1\}
\]

(ii) \(\Gamma_2 = (N, T, (P_0, S_0, R), (A_1, P_1), (A_2, P_2))\) where

\[
N = \{S_0, Q\}
\]

\[
T = \{a, b\}
\]

\[
P_0 = \{S_0 \rightarrow aS_0, S_0 \rightarrow aQ\}
\]

\(R\) is the language represented by the regular expression \((a^2)^* + (b^3)^*\)

\(Q\) is a query symbol

\[
P_1 = \{ab\}
\]

\[
A_1 = \{a, \lambda\}
\]

\[
P_2 = \{aaa\}
\]

\[
A_2 = \{b\}
\]

\[
(S_0, \{a, \lambda\}, \{b\}) \Rightarrow (aS_0, \{\lambda, a, a^2\}, \{bbb\}) \Rightarrow
\]

\[
(aaQ, \{\lambda, a, a^2, a^3, a^4\}, \{b^9\}) \vdash (a\{\lambda, a, a^4\}\{b^9\}, y_1, y_2)
\]

\[
L(\Gamma_2) = \{a^n\{\lambda, a^2, a^4, \ldots, a^{2n}\}b^{3n}/n \geq 1\}.
\]

The language generated by parallel communicating synchronized pure pattern grammar system with filter is denoted by \(CCPC(SPPL)\) and the language generated by parallel communicating synchronized pure pattern grammar by \(PC(SPPL)\). The corresponding families are denoted by \(CCPC(SPPL)\) and \(PC(SPPL)\) respectively.
Proposition 2.5.1

(1) \( \text{CCPC(SPPPL)} - \text{PC(SPPPL)} \neq \phi \)

(2) \( \text{CCPC(SPPPL)} \cap \text{PC(SPPPL)} \neq \phi \)

Proof.

(1) This is seen from the Example 2.5.1(ii). The language generated by \( \Gamma_2 \) is not a PC(SPPPL), because in a PC(SPPP) if a query symbol \( Q_j \) appears in the master component, then the string in the \( j^{th} \) component alone is communicated, as only one component can communicate to the master component at a time. But in \( \text{CCPC(SPPG)} \), if a query symbol \( Q \) appears in the master component, then the strings of the entire set of components are communicated, if they can pass through the filter and are concatenated with the string generated in the master component. Therefore \( \text{CCPC(SPPPL)} - \text{PC(SPPPL)} \neq \phi \).

(2) The result is seen from Example 2.5.1(i). The language \( L = \{ ba^n/n \geq 1 \} \) can be generated by \( \text{CCPC(SPPG)} \) and a PC(SPPP) \( \Gamma = (N, T, (P_0, S_0), (A_1, P_1)) \) where \( N = \{ S_0, Q, A \} \), \( T = \{ a, b \} \), \( P_0 = \{ S_0 \rightarrow bA, A \rightarrow aA, A \rightarrow aQ \} \), \( Q \) is a query symbol, \( P_1 = \{ ab \} \) and \( A_1 = \{ \lambda, a \} \).

Thus \( \text{CCPC(SPPPL)} \cap \text{PC(SPPPL)} \neq \phi \).

Proposition 2.5.2 The class \( \text{CCPC(SPPPL)} \) of languages in the returning mode coincides with the non-returning mode.
Proof. Let $\Gamma = (N, T, (P_0, S_0, R), (A_1, P_1), (A_2, P_2), \ldots, (A_n, P_n))$ where $(N, T, P_0, S_0)$ is a regular grammar and $R$ be a filter. The number of nonterminals in the right hand side of each rule in $P_0$ is one. Hence when the designated query symbol of the master component is replaced by a string after passing through the filter, we get a terminal word. No further generation is possible. Hence there is no difference between the returning and non-returning modes.

2.6 A Variant in Parallel Communicating Grammar Systems with Filters

In this section, a CCPC grammar system is considered with regular filters, regular or context-free master component and the other components are pattern or synchronized pure pattern or pure context-free grammars.

Example 2.6.1

(i) $\Gamma_1 = (N, T, (P_0, S_0, R), (A_1, P_1))$ where

$N = \{S_0, Q\}$

$Q$ is a query symbol

$T = \{a, b, c\}$

$P_0 = \{S_0 \rightarrow aS_0c, S_0 \rightarrow aQc\}$

$R$ is the language represented by the regular expression $b^*$

$A_1 = \{\lambda\}$

$P_1 = \{b\delta\}$

Here the master component is a context-free grammar and the other
component is a pattern grammar.

$$
\begin{align*}
\{S_0\}, \{\lambda\} & \Rightarrow \{aS_0c\}, \{b\} \Rightarrow \{a^2S_0c^2\}, \{b^2\} \Rightarrow^* \{a^nQc^n\}, \{b^n\} \vdash (a^nbc^n, \lambda) \\
L(\Gamma_1) &= \{a^nbc^n/n \geq 1\}
\end{align*}
$$

(ii) $\Gamma_2 = (N, T, (P_0, S_0, R), (A_1, P_1))$ where

$N = \{S_0, Q\}$

$Q$ is a query symbol

$T = \{a, b, c\}$

$P_0 = \{S_0 \to aS_0c, S_0 \to aQ\}$

$R$ is the language represented by the regular expression $b^+c$

$A_1 = \{c\}$

$P_1 = \{c \to bc\}$

Here the master component is a context-free grammar and the other is a pure context-free grammar.

$$
\begin{align*}
\{S_0\}, c & \Rightarrow \{aS_0c\}, \{bc\} \Rightarrow \{a^2S_0c^2\}, \{b^2c\} \Rightarrow^* \{a^nQc^{n-1}\}, \{b^nc\} \vdash (a^nb^ncc^n, \lambda) \\
L(\Gamma_2) &= \{a^nb^nc^n/n \geq 1\}
\end{align*}
$$

(iii) $\Gamma_3 = (N, T, (P_0, S_0, R), (A_1, P_1))$ where

$N = \{S_0, Q\}$

$Q$ is a query symbol

$T = \{a, b\}$

$P_0 = \{S_0 \to aS_0, S_0 \to Q\}$

$R$ is the language represented by the regular expression $ab^*$
\[ P_1 = \{ \delta b \} \]
\[ A_1 = \{ a \} \]
\[ (\{ S_0 \}, \{ a \}) \Rightarrow (\{ aS_0 \}, \{ ab \}) \Rightarrow (\{ a^2S_0 \}, \{ ab^2 \}) \Rightarrow^* (\{ a^{n-1}Q \}, \{ ab^n \}) \vdash (a^{n-1}\{ ab^n \}, \{ a \}) \]
\[ L(\Gamma_3) = \{ a^n b^n / n \geq 1 \}. \]

Here the master component is a regular grammar but the other component is a pattern grammar.

From the above examples the following results are obtained:

(i) In Example 2.6.1(i), the master component is context-free with regular filter and the other component is a pattern grammar but the language generated is a context sensitive language.

(ii) From the Example 2.6.1(ii), it is clear that a context-free master component with regular filter and a pure context-free grammar as the other component can also generate a context sensitive language.

(iii) In Example 2.6.1(iii), a system with regular master component, regular filter and a pattern grammar as the other component generates a context-free language.