Grammar Systems and Distributed Automata

With the need to solve different problems within a short time in an efficient manner, parallel and distributed computing have become essential. Study of such computations in the abstract sense, from the formal-language theory point of view, started with the development of grammar systems. In classical formal-language theory, a language is generated by a single grammar or accepted by a single automation. They model a single processor or we can say the devices are centralized. Though multi-tape Turing machines (TMs) try to introduce parallelism in a small way, the finite control of the machine is only one.
The introduction of distributed computing useful in analyzing computation in computer networks, distributed databases etc., has led to the notions such as distributed parallelism, concurrency, and communication. The theory of grammar systems and the distributed automata are formal models for distributed computing, where these notions could be formally defined and analyzed.
CD Grammar Systems

Definition

A CD grammar system of degree \( n \geq 1 \), is a construct:

\[
GS = (N, T, S, P_1, \ldots, P_n)
\]

Where \( N \) and \( T \) are disjoint alphabets (non terminals and terminals); \( S \in N \) is the start symbol and \( P_1, \ldots, P_n \) are the finite sets of rewriting rules over \( N \cup T \). \( P_1, \ldots, P_n \) are called components of the system.

Another way of specifying a CD grammar system is:

\[
GS = (N, T, S, G_1, \ldots, G_n)
\]

where \( G_i = (N, T, P_i, S), 1 \leq i \leq n \).
Definition

Let $GS = (N, T, S, P_1, \ldots, P_n)$ be a CD grammar system. We now define different protocols of co-operation.

1. Normal mode ($* \text{ mod } e$): $x \Rightarrow^{*}_{P_i}$ is defined by, $x \Rightarrow y$ without any restriction.

   The student works on the blackboard as long as he wants.

2. Terminating mode ($t \text{ mod } e$): for each $i \in \{1, \ldots, n\}$, the terminating derivation by the $i$th component, denoted by $x \Rightarrow^{t}_{P_i}$, is defined by $x \Rightarrow y$ if and only if $x \Rightarrow^{*}_{P_i} y$ and there is no $z \in (N \cup T)^*$ with $x \Rightarrow^{*}_{P_i} z$. 
3. \( = k \) mode: For each \( i \in \{1, \ldots, n\} \), the \( k \)-steps derivation by the \( i \)th component, denoted by \( P_i^{=k} \), is defined by \( x \Rightarrow_P y \) if and only if there are \( x_1, \ldots, x_{k+1} \in (N \cup T)^* \) such that \( x = x_i \), \( y = x_{k+1} \) and for each \( j \), \( 1 \leq j \leq k \)

\[
x_j^{P_i} \Rightarrow x_{j+1}.
\]

4. \( \leq k \) mode: For each component \( P_i \), the \( \leq k \) - steps derivation by the \( i \)th component denoted by \( P_i^{\leq k} \), is defined by:

\[
x \Rightarrow_P y \text{ if and only if } x \Rightarrow_P y \text{ for some } k' \leq k.
\]
5. \( \geq k \) mode: The \( \geq k \) steps of derivation by the \( i \)th component, denoted as \( \Rightarrow_{P_i}^{\geq k} \), is defined by

\[
x \Rightarrow_{P_i}^{\geq k} y \text{ if and only if } x \Rightarrow_{P_i}^{=k'} y \text{ for some } k' \geq k.
\]

Let \( D = \{*,t\} \cup \{\leq k, \geq k, = k \mid k \geq 1\} \).
Definition

The language generated by a CD grammar system $\mathcal{GS} = (N, T, S, P_1, \ldots, P_n)$ in derivation mode $f \in D$ is:

$$L_f(\mathcal{GS}) = \left\{ W \in T^* \mid S \xrightarrow[P_{i_1}^f]{} \alpha_1 \xrightarrow[P_{i_2}^f]{} \alpha_2 \ldots \xrightarrow[P_{i_m}^f]{} \alpha_m = w, m \geq 1, \quad 1 \leq i_j \leq n, 1 \leq j \leq m \right\}$$
Example

1. Consider the following CD grammar system:

\[ GS_1 = \left( \{S, X, X', Y, Y'\}, \{a, b, c\}, S, P_1, P_2 \right), \]

\[ P_1 = \left\{ S \to S, S \to XY, X' \to X, Y' \to Y \right\} \]

\[ P_2 = \left\{ X \to aX', Y \to bY'c, X \to a, Y \to bc \right\} \]

If \( f = * \) mode, the first component derives \( S \Rightarrow XY \) the second component derives from \( Y, bY'c \), it can switch to first component or derive \( aX' \) from \( X \).

In the first component \( X' \) can be changed to \( X \) or \( Y' \) can be changed to \( Y \) or both. The derivation proceeds similarly.
It is not difficult to see that the language generated is

$$\left\{ a^m b^n c^n \mid m, n \geq 1 \right\}.$$ 

The same will be true for

$$t \mod e = 1 \mod e, \geq 1 \mod e, \leq k \mod e \text{ for } k \geq 1.$$ 

But, if we consider $e = 2$, each component should execute two steps. In the first component $S \Rightarrow S \Rightarrow XY$. In the second component, $XY \Rightarrow aX'Y \Rightarrow aX'bY'c$.

Then control goes back to component one where $X'$ and $Y'$ are changed to $X$ and $Y$ in two steps. The derivation proceeds in the similar manner.
It is easy to see that the language generated by $GS_1$ in the $= 2$ mode is $\{a^n b^n c^n | n \geq 1\}$. A similar argument holds for $\geq 2$ – mode also and the language generated is the same.

At most, two steps of derivation can be done in each component. Hence, in the case of $= k$ or $\geq k$ mode where $k \geq 3$, the language generated is the empty set.
2. \( GS_2 = (\{S, A\}, \{a\}, S, P_1, P_2, P_3) \)

\[ P_1 = \{S \rightarrow AA\} \]

\[ P_2 = \{A \rightarrow S\} \]

\[ P_3 = \{A \rightarrow a\} \]

In the * mode \( \{a^n \mid n \geq 2\} \) is generated as the control can switch from component to component at any time.

A similar results holds for \( \leq 1, \leq k (k \geq 1) \) modes. For \( = 1, = k \), \( \geq k (k \geq 2) \), the language generate is empty as \( S \rightarrow AA \) can be used only once in \( P_1 \) and \( A \rightarrow a \) can be used once in \( P_3 \).
In the $t$ mode in $P_1$, $S \Rightarrow AA$ and if the control goes to $P_3$ from AA, $aa$ is derived. If the control goes to $P_2$ from AA, $SS$ is derived. Now the control has to go to $P_1$ to proceed with the derivation

$$SS \Rightarrow AAAAA,$$

and if the control goes to $P_2$, $S^4$ is derived; if it goes to $P_3$, $a^4$ is derived. It is easy to see that the language generated in $t$ mode is

$$\left\{ a^{2^n} \mid n \geq 1 \right\}.$$
3. \( GS_3 = \left( \{S, X_1, X_2\}, \{a, b\}, S, P_1, P_2, P_3 \right) \),

where

\[
P_1 = \{S \rightarrow S, S \rightarrow X_1X_1, X_2 \rightarrow X_1\}
\]

\[
P_2 = \{X_1 \rightarrow aX_2, X_1 \rightarrow a\}
\]

\[
P_3 = \{X_1 \rightarrow bX_2, X_1 \rightarrow b\}
\]

In \( ^* \) mode , \( = 1, \geq 1 \mod e, \leq k \mod e (k \geq 2) \), \( t \) mode the language generated will be \( \left\{ w | w \in \{a, b\}^*, |w| \geq 2 \right\} \). In \( = 2 \) mode , each component has to execute two steps , so the language generated will be \( \left\{ ww | w \in \{a, b\}^+ \right\} \).
A similar argument holds for $\geq 2$ steps. For $= k or \geq k \mod es(k \geq 3)$, the language generated is empty, as each component can use at most two steps before transferring the control.

We state some results about the generative power without giving proof. The proofs are fairly simple, and can be tried as exercise. It can be easily seen that for CD grammars systems working in any of the modes defined having regular, linear, context–sensitive, or type 0 components, respectively, the generative power does not change; i.e., they generate the families of regular, linear, context–sensitive, or recursively enumerable languages, respectively.
But by the example given, we find that CD grammar systems with context–free components can generate context–sensitive languages. Let $CD_n(f)$ denote the family of languages generated by CD grammar systems with $\varepsilon$–free context free components, the number of components being at most $n$. When the number of components is not limited, the family is denoted by $CD_\infty(f)$ if $\varepsilon$–rules are allowed. The corresponding families are denoted by $CD_n^\varepsilon(f)$ and $CD_\infty^\varepsilon(f)$, respectively.
PC Grammar systems

Definition
A PC grammar system of degree \( n \), \( n \geq 1 \), is an \((n + 3)\)–tuple:

\[
GP = \left( N, K, T, (S_1, P_1), \ldots, (S_n, P_n) \right),
\]

Where \( N \) is a non-terminal alphabet, \( T \) is a terminal alphabet, \( K = \{Q_1, Q_2, \ldots, Q_n\} \) are query symbols. \( N, T, K \) are mutually disjoint. \( P_i \) is a finite set of rewriting rules over \( N \cup K \cup T \), and \( S_i \in N \) for all \( 1 \leq i \leq n \).

Let \( V_{GP} = N \cup K \cup T \)
The sets $P_i$ are called components of the system. The index $i$ of $Q_i$ points to the $i$th component $P_i$ of GP.

An equivalent representation for GP is $(N, K, T, G_1, \ldots, G_n)$, where $G_i = (N \cup K, T, S_i, P_i), 1 \leq i \leq n$. 

Definition

Given a PC grammar system

\[ GP = \left( N, K, T, (S_1, P_1), \ldots, (S_n, P_n) \right) \]

as above for two \( n \) – tuples \((x_1, x_2, \ldots, x_n), (y_1, \ldots, y_n)\), with \( x_i, y_i \in V_{GP}^* \), \( 1 \leq i \leq n \), where \( x_1 \not\in T^* \), we write

\( (x_1, \ldots, x_n) \Rightarrow (y_1, \ldots, y_n) \)

if one of the two following two cases holds.

1. For each \( i, 1 \leq i \leq n \), \( x_i \mid_K = 0 \) (i.e., no query symbol in \( x_i \) ), and either \( x_i \Rightarrow y_i \) by a rule in \( P_i \) or \( x_i = y_i \in T^* \).

2. There is \( i, 1 \leq i \leq n \), such that \( x_i \mid_K > 0 \) (i.e., \( x_i \) has query symbols). Let for each such \( i, x_i = z_1 Q_i z_2 Q_i \ldots z_t Q_i z_{t+1}, t \geq 1 \) for \( z_i \in (N \cup T)^* \), \( 1 \leq j \leq t + 1 \).
If $|x_{ij}|_K = 0$, for all $j$, $1 \leq j \leq t$, then $y_i = z_1 x_1 z_2 x_2 \cdots z_t x_t z_{t+1}$

and $y_{ij} = S_j$

( in returning mode ) and $y_{ij} = x_{ij}$ ( in non returning mode ) $1 \leq j \leq t$.

if for some $j$, $1 \leq j \leq t$, $|x_{ij}|_k \neq 0$, then $y_i = x_i$. For all $i$, $1 \leq i \leq n$,

such that $y_i$ is not specified above, we have $y_i = x_i$.

An n-tuple $(x_1, \ldots, x_n)$ with $x_i \in V_{GP}^*$ is called an instantaneous description (ID) (of GP).
Thus an ID \( (x_1, \ldots, x_n) \) directly gives rise to an ID \( (y_1, \ldots, y_n) \), if either

(a) Component wise derivation: No query symbol appears in \( x_1, \ldots, x_n \); then we have \( x_i \Rightarrow y_i \) using a rule in \( P_i \), if \( x_i \) has a non terminal. If \( x_i \in T^* \), \( y_i = x_i \).

(b) Communication step: Query symbols occur in some \( x_i \). Then, a communication step is performed. Each occurrence of \( Q_j \) in \( x_i \) is replaced by \( x_j \), provided \( x_j \) does not contain query symbols. In essence a component \( x_i \) containing query symbols is modified only when all occurrences of query symbols in it refer to strings without occurrences of query symbols.
In this communication step \( x_j \) replaces the query symbol \( Q_j \). After that, GS resumes starting from axiom (this is called returning mode) or continues from where it was (this is called non-returning mode). Communication has priority over rewriting. No rewriting is possible as long as one query symbol is present in any component. If some query symbols in a component cannot be replaced in a given communication step, it may be possible that they can be replaced in the next step. When the first component (master) has a terminal string derivation stops. \( \Rightarrow \) is used to represent both rewriting and communication steps. \( \ast \Rightarrow \) is the reflexive transitive closure of \( \Rightarrow \). We write \( \Rightarrow^r \) for returning mode \( \Rightarrow^{nr} \) for non-returning mode.
Definition

The language generated by a PC grammar system GP

1. In returning mode is:

\[ L_r(GP) \left\{ x \in T^* \mid (S_1, \ldots, S_n) \implies_r (x, \alpha_2, \ldots, \alpha_n), \alpha_i, V_{GP}^*, 2 \leq i \leq n \right\} \]

2. In non-returning mode is:

\[ L_{nr}(GP) \left\{ x \in T^* \mid (S_1, \ldots, S_n) \implies_{nr} (x, \alpha_2, \ldots, \alpha_n), \alpha_i \in V_{GP}^*, 2 \leq i \leq n \right\} \]

If a query symbol is present, rewriting is not possible in any component. If circular query occurs, communication will not be possible and the derivation halts without producing a string for the language.
Circular query means something like the following:

component i has query symbol $Q_j$

component j has query symbol $Q_k$

and component k has query symbol $Q_i$

this is an example of a cycle query

The first component is called the master and the language consists of terminal strings derived there.

Generally, any component can introduce the query symbols. This is called non centralized system. If only the first component is allowed to introduce query symbols, it is called a centralized system.
A PC grammar system is said to be regular, linear, context free, and context sensitive, depending on the type of rules used in components.

There are a number of results on the hierarchy of PC grammar systems. Csuhaj Varju et al., (1997) gave a detailed description of these systems.
Example

1. Let

\[ GP_{1} = \left\{ \{S_1, S'_1, S_2, S_3\}, \{Q_1, Q_2, Q_3\}, \{a, b\}, (S_1, P_1), (S_2, P_2), (S_3, P_3) \right\} \]

\( P_1 \) Consists of rules:

1. \( S_1 \rightarrow abc \)
2. \( S_1 \rightarrow a^2 b^2 c^2 \)
3. \( S_1 \rightarrow aS'_1 \)
4. \( S_1 \rightarrow a^3 Q_2 \)
5. \( S'_1 \rightarrow aS \)
6. $S'_1 \rightarrow a^3Q_2$

7. $S_2 \rightarrow b^2Q_3$

8. $S_3 \rightarrow c$

$P_2 = \{S_2 \rightarrow bS_2\}$

$P_3 = \{S_3 \rightarrow cS_3\}$

$L_r(GP_1) = L_{nr}(GP_1) = \{a^n b^n c^n \mid n \geq 1\}$
This can be seen as follows

\[
\begin{array}{ccc}
\frac{G_1}{S_1} & \frac{G_2}{S_2} & \frac{G_3}{S_3} \\
abc & bS_2 & cS_3 \\
\end{array}
\]

derivation stops \( abc \in L_r \left( GP_1 \right), L_{nr} \left( GP_1 \right) \)

\[
\begin{array}{ccc}
S_1 & S_2 & S_3 \\
 a^2b^2c^2 & bS_2 & cS_3 \\
\end{array}
\]

\( a^2b^2c^2 \in L_r \left( GP_1 \right), L_{nr} \left( GP_1 \right) \)
\[
\begin{array}{ccc}
G_1 & G_2 & G_3 \\
\frac{S_1}{S_1} & \frac{S_2}{S_2} & \frac{S_3}{S_3} \\
\frac{a^3 Q_2}{bS_2} & \frac{bS_2}{S_2} & \frac{cS_3}{cS_3} \\
\frac{a^3 bS_2}{a^3 bS_2} & \frac{S_2}{S_2} & \frac{cS_3}{cS_3} \\
\frac{a^3 bb^2 Q_3}{bS_2} & \frac{bS_2}{bS_2} & \frac{c^2 S_3}{c^2 S_3} \\
\frac{a^3 b^3 c^2 S_3}{bS_2} & \frac{bS_2}{bS_2} & \frac{S_3}{S_3} \\
\frac{a^3 b^3 c^3}{b^2 S_2} & \frac{b^2 S_2}{b^2 S_2} & \frac{cS_3}{cS_3} \text{ in returning mode}
\end{array}
\]
\begin{align*}
S_1 & \quad S_2 & \quad S_3 \\
\quad a^3Q_2 & \quad bS_2 & \quad cS_3 \\
\quad a^3bS_2 & \quad bS_2 & \quad cS_3 \\
\quad a^3bb^2Q_3 & \quad b^2S_2 & \quad c^2S_3 \\
\quad a^3b^3c^2S_3 & \quad b^2S_2 & \quad c^2S_3 \\
\quad a^3b^3c^3 & \quad b^3S_2 & \quad c^3S_3 \quad \text{in non returning mode}
\end{align*}
$$a^3b^3c^3 \in L_r(GP_1), L_{nr}(GP_1)$$

<table>
<thead>
<tr>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$S_2$</td>
<td>$S_3$</td>
</tr>
<tr>
<td>$aS_1$</td>
<td>$bS_2$</td>
<td>$cS_3$</td>
</tr>
<tr>
<td>$aa^3Q_3$</td>
<td>$b^2S_2$</td>
<td>$c^2S_3$</td>
</tr>
<tr>
<td>$a^4b^2S_2$</td>
<td>$S_2$</td>
<td>$c^2S_3$</td>
</tr>
<tr>
<td>$a^4b^2b^2Q_3$</td>
<td>$bS_2$</td>
<td>$c^3S_3$</td>
</tr>
<tr>
<td>$a^4b^2c^3c_3$</td>
<td>$bS_2$</td>
<td>$S_3$</td>
</tr>
</tbody>
</table>
| $a^4b^4c^4$ | $b^2S_2$ | $cS_3$  
in returning mode |
\[
\begin{array}{ccc}
S_1 & S_2 & S_3 \\
\ aS_1 & bS_2 & cS_3 \\
\ aa^3Q_2 & b^2S_2 & c^2S_3 \\
\ a^4b^2S_2 & b^2S_2 & c^2S_3 \\
\ a^4b^2b^2Q_3 & b^3S_2 & c^3S_3 \quad \text{in non returning mode} \\
\ a^4b^4c^3S & b^3S_2 & c^3S_3 \\
\ a^4b^4c^4 & b^4S_2 & c^4S_3 \\
\end{array}
\]

\[a^4b^4c^4 \in L_1(GP_1), L_{nr}(GP_1).\]
2. Let $GP_2 = \left( \{S_1, S_2\}, \{Q_1, Q_2\}, \{a, b\}, (S_1, P_1)(S_2, P_2) \right)$, where

$$P_1 = \{S_1 \to S_1, S_1 \to Q_1 Q_2\}$$

$$P_2 = \{S_2 \to aS_2, S_2 \to bS_2, S_2 \to a, S_2 \to b\}$$

$$L_r(GP_2) = L_{nr}(GP_2) = \left\{ w w \mid w \in \{a, b\}^+ \right\}$$
We see how abbabb is derived.

\[
\begin{array}{c}
G_1 \\
S_1 \\
S_1 \\
S_1 \\
Q_2Q_2 \\
\text{abbabb}
\end{array}
\begin{array}{c}
G_2 \\
S_2 \\
aS_2 \\
abS_2 \\
\text{abb} \\
\text{in returning mode}
\end{array}
\begin{array}{c}
S_2 \\
\text{abb} \\
\text{in non returning mode}
\end{array}
\]

However, \( abbabb \in L_r \left( GP_1 \right), L_{nr} \left( GP_1 \right) \)

This type of communication is called communication by request. There is also another way of communication known as communication by command. (Dassow and Paun, 1997). A restricted version of this communication by command is found useful in characterizing the workload in computer networks (Arthi et al., 2001).