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 ≥ 1, is a construct.:

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

Where N and T are disjoint alphabets (non terminals and terminals) ;

$S \in N$ is the start symbol and $P_1, \ldots\ldots. P_n$ are the finite sets of rewriting rules over $N \cup T$. $P_1, \ldots\ldots. P_n$ are called components of the system.

Another way of specifying a CD grammar system is :

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

where $G_i = (N, T, P_i, S), 1 \le i \le n.$

Definition

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

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

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

2. Terminating mode $(t\ \text{mode})$: for each $i \in \{1, \ldots\ldots n\}$, the terminating derivation by the ith component , denoted by $\overset{t}{\underset{P_i}{\Rightarrow}}$, is defined by $x \overset{t}{\underset{P_i}{\Rightarrow}} y$ *if* and only *if* $x \overset{*}{\underset{P_i}{\Rightarrow}} y$ and there is no $z \in (N \cup T)^*$ with $y \underset{P_i}{\Rightarrow} z$.

3 . = k mode : For each $i \in \{1,......,n\}$ the k-steps derivation by the ith component, denoted by $\overset{=k}{\underset{P_i}{\Rightarrow}}$, is defined by $x \overset{=k}{\underset{P_i}{\Rightarrow}} y$ if and only if there are $x_1,.......x_{k+1} \in (N \cup T)^*$ such that $x = x_i$, $y = x_{k+1}$ and for each j , $1 \leq j \leq k$

$$x_j \underset{P_i}{\Rightarrow} x_{j+1}.$$

4. $\leq$ k mode : For each component P_i, the $\leq$ k – steps derivation by the ith component denoted by $\overset{\leq k}{\underset{P_i}{\Rightarrow}}$, is defined by :

$$x \overset{\leq k}{\underset{P_i}{\Rightarrow}} y \text{ if and only if } x \overset{=k'}{\underset{P_i}{\Rightarrow}} y \text{ for some } k' \leq k.$$

5. $\geq$ k mode : The $\geq$ k steps of derivation by the ith component , denoted as $\underset{P_i}{\overset{\geq k}{\Rightarrow}}$, is defined by

$$x \underset{P_i}{\overset{\geq k}{\Rightarrow}} y \text{ if and only if } x \underset{P_i}{\overset{=k'}{\Rightarrow}} y \text{ for some } k' \geq k.$$

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

Definition

The language generated by a CD grammar system

$GS = \left(N, T, S, P_1, \ldots\ldots, P_n\right)$ in derivation mode $f \in D$ is :

$$L_f\left(GS\right) = \left\{ \begin{array}{r} W \in T^* \mid S \underset{P_{i_1}}{\overset{f}{\Rightarrow}} \alpha_1 \underset{P_{i_2}}{\overset{f}{\Rightarrow}} \alpha_2 \ldots\ldots \underset{P_{i_m}}{\overset{f}{\Rightarrow}} \alpha_m = w, m \geq 1, \\ 1 \leq i_j \leq n, 1 \leq j \leq m \end{array} \right\}$$

Example

1. Consider the following CD grammar system :

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

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

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

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

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

The same will be true for

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

But, if we consider = 2 mode , 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 $\{a^n b^n c^n \mid n \geq 1\}$. A similar argument holds for ≥ 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 ≥ k mode where k ≥ 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 $\geq 1, \leq k\,(k \geq 1)$ 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

$SS \Rightarrow AAAA$, 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

$$P_1 = \{S \to S, S \to X_1 X_1, X_2 \to X_1\}$$

$$P_2 = \{X_1 \to aX_2, X_1 \to a\}$$

$$P_3 = \{X_1 \to bX_2, X_1 \to b\}$$

In * mode , $=1, \geq 1$ mod$e$, $\leq k$ mod$e(k \geq 2)$, t mode the language generated will be $\left\{w \mid w \in \{a,b\}^*, |w| \geq 2\right\}$. In = 2 mode , each component has to execute two steps , so the language generated will be $\left\{ww \mid w \in \{a,b\}^+\right\}$.

A similar argument holds for ≥ 2 steps . For $= k$ or $\geq k$ modes $(k \geq 3)$, 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 ≥ 1 , is an (n + 3) – tuple :

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

Where N is a non - terminal alphabet , T is a terminal alphabet ,

$K = \{Q_1, Q_2, \ldots\ldots\ldots Q_n\}$ 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 $1 \le i \le n$.

$$Let\ V_{GP} = N \cup K \cup T$$

The sets P_i are called components of the system. The index i of Q_i points to the ith component P_i of GP .

An equivalent representation for GP is $\left(N, K, T, G_1, \ldots\ldots\ldots, G_n\right)$,

where $G_1 = \left(N \cup K, T, S_i, P_i\right), 1 \le i \le n$.

Definition

Given a PC grammar system

$GP = \left(N, K, T, (S_1, P_1).....(S_n, P_n)\right)$,

as above for two n – tuples $(x_1, x_2, ..., x_n)$, $(y_1, ...y_n)$, with $x_i, y_i \in V_{GP}^{*}, 1 \le i \le n$, where $x_1 \notin T^{*}$, we write $(x_1, ..., x_n) \Rightarrow (y_1, ...y_n)$

if one of the two following two cases holds.

1. For each i ,$1 \le i \le n, |x_i|_K = 0$ (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 ,$1 \le i \le n$, such that $|x_i|_K > 0$. (i.e., x_i has query symbols). Let for each such i ,$x_i = z_1 Q_{i_1} z_2 Q_{i_2} ....z_t Q_{i_t} z_{t+1}$,$t \ge 1$ for

$z_i \in (N \cup T)^{*}, 1 \le j \le t+1.$

If $|x_{i_j}|_K = 0$, for all j , $1 \le j \le t$, then $y_i = z_i x_{i_1} z_2 x_{i_2} \ldots. z_t x_{i_t} z_{t+1}$

and $y_{i_j} = S_j$

(in returning mode) and $y_{i_j} = x_{i_j}$ (in non returning mode) $1 \le j \le t$.
if for some j $, 1 \le j \le t, |x_{i_j}|_k \ne 0$, then $y_i = x_i$. For all i , $1 \le i \le n$,

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

An n-tuple $(x_1, \ldots\ldots, x_n)$ with $x_i \in V_{GP}^*$ is called an instantaneous description (ID) (of GP).

Thus an ID $(x_1,\ldots\ldots,x_n)$ directly gives rise to an ID $(y_1,\ldots\ldots y_n)$,if either

(a)Component wise derivation: No query symbol appears in $x_1,\ldots\ldots,x_n$; 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 . $\overset{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$. We write $\underset{r}{\Rightarrow}$ for returning mode $\underset{nr}{\Rightarrow}$ for non returning mode .

Definition

The language generated by a PC grammar system GP

1. In returning mode is :

$$L_r(GP)\left\{x \in T^* \mid (S_1,\ldots,S_n) \overset{*}{\underset{r}{\Rightarrow}} (x,\alpha_2,\ldots.,\alpha_n),\alpha_i,V_{GP}^*,2 \le i \le n\right\}$$

2. In non - returning mode is :

$$L_{nr}(GP)\left\{x \in T^* \mid (S_1,\ldots,S_n) \overset{*}{\underset{nr}{\Rightarrow}} (x,\alpha_2,\ldots.,\alpha_n),\alpha_i \in V_{GP}^*,2 \le i \le n\right\}$$

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

$$GP_1 = \begin{pmatrix} \{S_1, S_1', S_2, S_3\}, \{Q_1, Q_2, Q_3\}, \{a, b\}, (S_1, P_1), (S_2, P_2), \\ (S_3, P_3) \end{pmatrix}$$

P_1 Consists of rules :

1. $S_1 \to abc$
2. $S_1 \to a^2b^2c^2$
3. $S_1 \to aS_1'$
4. $S_1 \to a^3Q_2$
5. $S_1' \to aS$

6. $S_1' \to a^3 Q_2$

7. $S_2 \to b^2 Q_3$

8. $S_3 \to c$

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

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

$$L_r(GP_1) = L_{nr}(GP_1) = \{a^n b^n c^n \mid n \geq 1\}$$

This can be seen as follows

$\frac{G_1}{S_1}$	$\frac{G_2}{S_2}$	$\frac{G_3}{S_3}$
abc	bS_2	cS_3

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

S_1	S_2	S_3
$a^2b^2c^2$	bS_2	cS_3

$$a^2b^2c^2 \in L_r\left(GP_1\right), L_{nr}\left(GP_1\right)$$

$\frac{G_1}{S_1}$	$\frac{G_2}{S_2}$	$\frac{G_3}{S_3}$	
a^3Q_2	bS_2	cS_3	
a^3bS_2	S_2	cS_3	
$a^3bb^2Q_3$	bS_2	c^2S_3	
$a^3b^3c^2S_3$	bS_2	S_3	
$a^3b^3c^3$	b^2S_2	cS_3	in returning mode

S_1	S_2	S_3	
a^3Q_2	bS_2	cS_3	
a^3bS_2	bS_2	cS_3	
$a^3bb^2Q_3$	b^2S_2	c^2S_3	
$a^3b^3c^2S_3$	b^2S_2	c^2S_3	
$a^3b^3c^3$	b^3S_2	c^3S_3	in non returning mode

$a^3b^3c^3 \in L_r\left(GP_1\right), L_{nr}\left(GP_1\right)$

$\frac{G_1}{S_1}$	$\frac{G_2}{S_2}$	$\frac{G_3}{S_3}$	
aS_1	bS_2	cS_3	
aa^3Q_3	b^2S_2	c^2S_3	
$a^4b^2S_2$	S_2	c^2S_3	
$a^4b^2b^2Q_3$	bS_2	c^3S_3	
$a^4b^2c^3c_3$	bS_2	S_3	
$a^4b^4c^4$	b^2S_2	cS_3	in returning mode

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	in non returning mode
$a^4b^4c^3S_3$	b^3S_2	c^3S_3	
$a^4b^4c^4$	b^4S_2	c^4S_3	

$$a^4b^4c^4 \in L_1\left(GP_1\right), L_{nr}\left(GP_1\right).$$

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

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

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

$$L_r(GP_2) = L_{nr}(GP_2) = \left\{ ww \mid w \in \{a, b\}^+ \right\}$$

We see how abbabb is derived .

$\frac{G_1}{S_1}$	$\frac{G_2}{S_2}$	
S_1	aS_2	
S_1	abS_2	
Q_2Q_2	abb	
$abbabb$	S_2	in returning mode
$abbabb$	abb	in non returning mode

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).