Theory of Computation: CS-202

Regular Grammars & Languages

Outline

- ❑Regular Grammars
- ❑Regular Languages

Regular Grammar (RG)

A third way to describe Regular Language (RL) is by means of RG.

- Right Linear Grammar
- Left Linear Grammar

Right Linear Grammar

A Grammar $G = (V, T, S, P)$ is said to be Right Linear if all productions are of the form

$$A \rightarrow nB$$

or $\quad A \rightarrow n$

where, A & $B \in V$ and $n \in T^*$

Left Linear Grammar

A Grammar $G = (V, T, S, P)$ is said to be Left Linear if all productions are of the form

$$A \rightarrow Bn$$

$$or \qquad A \rightarrow n$$

where, A & $B \in V$ and $n \in T^*$

cont…

- A Regular Grammar is either right linear or left linear

Note:

- In RG at most one variable appear on the R.H.S. of any production and that variable must be either right most or left most in the production.
- A language *L* is called regular if there is a finite automata.

Example

$$G_1 = (\{S\}, \{a, b\}, S, P_1)$$

P_1 given as

$S \rightarrow abS \mid a$ is right linear.

$$G_2 = (\{S, S_{1,} S_2\}, \{a, b\}, S, P_2)$$

P_2 is given as

$$S \rightarrow S_1 ab$$

$$S_1 \rightarrow S_1 ab \mid S_2$$

$S_2 \rightarrow a$ is left linear.

Both G_1 & G_2 are RG.

Cont…

- For G_1

$$S \rightarrow abS \rightarrow ababS \rightarrow aaaba$$

is a derivation

$$L(G_1) = L((ab)^*a)$$

- For G_2

$S \rightarrow S_1 ab \rightarrow S_1\text{abab} \rightarrow S_2\text{abab} \rightarrow \text{aabab}$

R.E. $\rightarrow$ a$(ab)^*$

$$L(G_2) = L(a(ab)^*)$$

The $G = (\{S, A, B\}, \{a, b\}, S, P)$

with production

$$S \rightarrow A$$

$A \rightarrow aB \mid \lambda$

$$B \rightarrow Ab$$

is not regular, although the production are either right linear or left linear.

- ➢ The Grammar itself is neither right linear & left linear.
- ➢ This language is linear language

In linear language, at most one variable can occur on R.H.S. of productions without restriction on the position of this variable.

- ❖ Every RG is linear but not all linear grammar are regular.

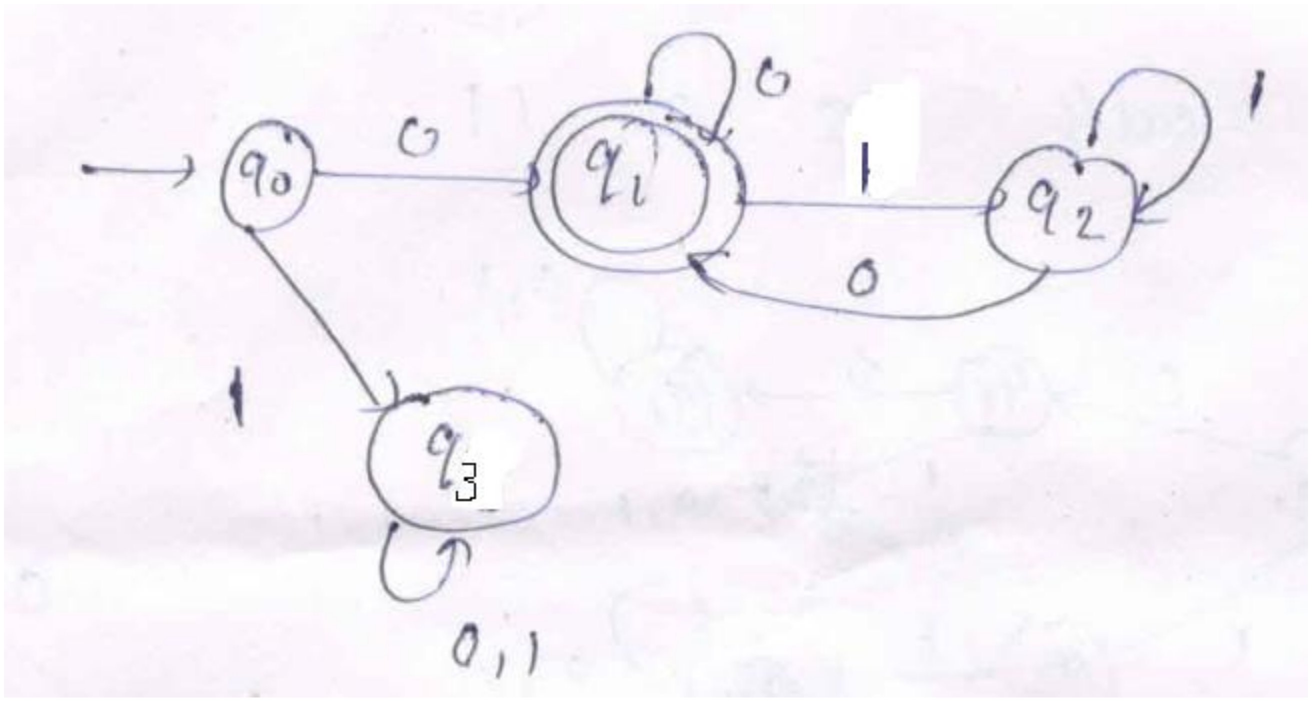
Pumping Lemma for Regular Language

- The Pumping Lemma for regular languages is based on the idea that regular languages can be recognized by finite automata.

- The intuition behind the Pumping Lemma lies in the fact that if a language is regular, there is a finite automaton that recognizes it.

- This automaton has a finite number of states, and when it processes a string longer than the number of states, it must revisit a state, leading to repetition.

- Draw a DFA which accepts all the strings on $\Sigma=\{0,1\}$ which must start & end with '0'.



Pumping Lemma for Regular Language

- It is used to prove that the given language is not Regular.
- It can't be used to prove that a language is Regular.

Let L be infinite RL then there exists some positive integer M $s.t.$ any $W \epsilon L$ with $|W| \geq M$ can be decomposed as

$$W = XYZ$$

with

$$|XY| \leq M$$
$$|Y| \geq 1$$

s.t. $W_i = XY^iZ$ is also in L $for\ i = 0, 1, 2, \ldots$

Note: Pumping lemma is a negative test which can tell that the given language is not Regular, but from the definition it is not telling that if the conditions satisfy then the language will be Regular.

- Let L is Regular
- It has a pumping length 'M'
- All strings longer than M can be pumped $|W| \geq M$
- Find W, $|W| \geq M$
- Divide W into XYZ
- Show that $XY^iZ \notin L$ for some i
- Consider all ways that W can be divided in XYZ
- Show that none of the string satisfy the pumping condition at the same time

$$\rightarrow \textit{W can't be pumped} \Rightarrow \textit{contradiction}$$

Example:

L={$a^n b^n$ | $n \geq 0$} is not regular, we can prove it by contradiction.

Let *L* is regular, then $a^m b^m$has a pumping length *M.*

- *L* can be divided into *XYZ*

$$W = aaaaabbbbb$$

take *M* sufficiently large.

take *M* = *5*

Y can be in 'a' , 'b' or 'ab' part

take i=2, XY^2Z

$$\underbrace{aaa}_{x}\underbrace{aa}_{y}\underbrace{bbbbb}_{z} \longrightarrow \quad |XY| = 5, \qquad Z = 5$$

$$\underbrace{aaaa}_{x}\underbrace{abbb}_{y}\underbrace{bb}_{z} \longrightarrow \quad |XY| = 8, \qquad Z = 2$$

$$\underbrace{aaa}_{x}\underbrace{aabb}_{y}\underbrace{bbb}_{z} \longrightarrow \quad |XY| = 7, \qquad Z = 3$$

All the three conditions are not satisfied .

L is not regular.

Closure Properties of Regular Languages

Closure Properties of Regular Languages

Theorem: If L_1 and L_2 are Regular language then $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 L_2$, $\overline{L_1}$ & ${L_1}^*$ are also Regular languages.

We say that the family of RL is closed under union, intersection, concatenation, complementation and star closure.

Proof:

If $L_1 and\ L_2$ are RL then there exist RE r_1 and r_2

s.t. $L_1 = L(r_1)$ and $L_2 = L(r_2)$

By definition, $r_1 + r_2, r_1 r_2$ and ${r_1}^*$ are RE denoting languages $L_1 \cup L_2, L_1 L_2$ and ${L_1}^*$.

Thus, family of RL is closed under union, concatenation & star closure.

To show closure under complementation

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts L_1.

then the DFA

$$\bar{M} = (Q, \Sigma, \delta, q_0, Q - F) \text{ accepts } \overline{L_1}.$$

For the given string w $\in \Sigma^*$

$$\delta^*(q_0, w) \in F, \qquad \rightarrow w \in L_1$$

$$\delta^*(q_0, w) \in Q - F, \qquad \rightarrow w \in \overline{L_1}$$

Closure under intersection-

Let $L_1 = L(M_1)$ and $L_2 = L(M_2)$

Where, $M_1 = (Q, \Sigma, \delta, q_0, F_1)$

$M_2 = (P, \Sigma, \delta, p_0, F_2)$ are DFA's.

We can construct from M_1 and M_2 a combined automation

$\widehat{M} = (\hat{Q}, \Sigma, \hat{\delta}, (q_0, p_0), \hat{F})$

Where $\hat{Q} = Q * P$ consist of pairs (q_i, p_j) whenever M_1 is in state q_i & M_2 is in state p_j

$\hat{\delta}(q_i, p_j, a) = (q_k, p_l)$

Whenever $\delta_1(q_i, a) = q_k$

$$\delta_2(p_j, a) = p_l$$

$\hat{F}$ is defined as the set of all (q_i, p_j)

s.t. $q_i \in F_1$ & $p_j \in F_2$

$w \in L_1 \cap L_2$ iff it is accepted by $\widehat{M}$

$L_1 \cap L_2$ is regular.

Closure under other operations-

Suppose Σ and T are alphabets, then a f^n

$$h : \Sigma \rightarrow T^*$$

iscalled homomorphism. homomorphism is a substitution in which a single letter is replaced with a string.

If $w = a_1 a_2 ... a_n$

then $h(w) = h(a_1)\, h(a_2) ... h(a_n)$

If L is a language on Σ, *then* its homomorphic image is defined as

$$h(L) = \{h(w) : w \in L\}$$

Example:

Let Σ= {a,b}, Γ = {a,b,c} and

h is defined by h(a) = ab

h(b) = bbc

h(ab) = h(a) h(b)

= abbbc

& if L={aa, aba}

homomorphism image of L is

h(L) = {abab, abbbcab}

Thank you