UNIT-II

REGULAR EXPRESSIONS

The languages accepted by finite automata are easily described by simple expressions called regular expressions.

Let Σ be an alphabet. The regular expression over Σ and the sets that they denote are defined recursively as follows:

Rules:

1. ϕ is a regular expression and denotes the empty set.
2. ϵ is a regular expression and denotes the set $\{\epsilon\}$.
3. For each 'a' in Σ, 'a' is a regular expression and denotes the set $\{a\}$
4. If 'r' and 's' are regular expressions denoting the languages 'R' and 'S', respectively, then $(r+s)$, (rs), and (r^*) are regular expressions that denote the sets $R \cup S$, RS and R^*, respectively.
5. If 'r' is a regular expression, then (r) is also a regular expression

Regular expression	Regular set
00	{00}
$(0+1)^*$	$\{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\}$
$(0+1)^* 00 (0+1)^*$	set of all strings with two consecutive
$(1+10)^*$	strings beginning with '1' and not having two consecutive 0's.
$(0+1)^* 011$	set of all strings ending in 011.
$0^* 1^* 2^*$	any no. of 0's followed by any no. of 1's followed by a

Regular set: Any set represented by a regular expression is called a regular set.

Ex.

a	denotes the regular set	{a}
a+b	"	{a, b}
ab	"	{ab}
a^*	"	$\{\epsilon, a, aa, aaa, aaaa, \ldots\}$
$(a+b)^*$	"	$\{\epsilon, a, b, aa, ab, ba, bb, \ldots\}$

Regular ~~Expression~~	Regular set (or) language
① ϵ	$\{\epsilon\}$
② 0	$\{0\}$
③ 001	$\{001\}$
④ 0+1	$\{0,1\}$
⑤ 0+10	$\{0,10\}$
⑥ $(1+\epsilon)001$	$\{1,\epsilon\}\{001\}$
⑦ $\epsilon+ab$	$\{\epsilon,ab\}$
⑧ 0^*	$\{\epsilon,0,00,000,\ldots\}$
⑨ 00^*	$\{0,00,000\ldots\}$
⑩ $(0+1)^*$	{ All strings of 0's and 1's }
⑪ $(0+1)^*00$	{ All strings of 0's and 1's ending with 00 }
⑫ $(0+1)^*00(0+1)^*$	{ All strings of 0's and 1's at least two consecutive 0's }
⑬ $a^*b^*c^*$	{ any no of a's followed by b's followed by c's }

~~Applications~~

① Lexical Analysis
② search engines.
③ word processors.
④ Text editor.

Identity rules for regular expressions

I_1: $\phi + R = R$

I_2: $\phi R = R\phi = \phi$

I_3: $\epsilon R = R\epsilon = R$

I_4: $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$

I_5: $R + R = R$

I_6: $R^* R^* = R^*$

I_7: $RR^* = R^* R$

I_8: $(R^*)^* = R^*$

I_9: $\epsilon + RR^* = \epsilon + R^* R = R^*$

I_{10}: $(PQ)^* P = P(QP)^*$

I_{11}: $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$

I_{12}: $(P+Q)R = PR + QR$ and $R(P+Q) = RP + RQ$

(Q) Give an regular expression for representing the set L of strings in which every 0 is immediately followed by atleast two 1's.

Sol: If $w \in L$ then

(i) w does not contain any '0' (or)

(ii) it contains a 0 followed by 11.

$$(1+011)^* = \{\epsilon, 1, 011, 11, 1011, 0111, 011011 \cdots\}$$

(Q) Prove $R = \epsilon + 1^*(011)^*(1^*(011)^*)^* = (1+011)^*$

Sol:

LHS: $= \epsilon + 1^*(011)^*(1^*(011)^*)^*$

$= (1^*(011)^*)^* \quad \{ \epsilon + RR^* = R^* \}$

$= (1+011)^* \quad ((P^*Q^*)^* = (P+Q)^*)$

$=$ RHS

(Q) Prove $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$

$= 0^*1(0+10^*1)^*$

LHS: $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$

$= (1+00^*1)(\epsilon + (0+10^*1)^*(0+10^*1)) \quad [$

$= (1+00^*1)(0+10^*1)^* \quad [\because \epsilon + RR^* = R^*]$

$= (\epsilon+00^*)1(0+10^*1)^*$

$= 0^*1\cdot(0+10^*1)^* =$ RHS

(8) simplify the folbowing regular expression

$(0+11^*0)+(0+11^*0)(10+10^*1)^*(10+10^*1)$

Sol: $(0+11^*0)+(0+11^*0)(10+10^*1)^*(10+10^*1)$

$= (0+11^*0)(\epsilon+(10+10^*1)^*(10+10^*1))$

$= (0+11^*0)(10+10^*1)^*$ $\left[\because \epsilon+R^*R=R^*\right]$

$= (\epsilon+11^*)0(10+10^*1)^*$

$= 1^*0(10+10^*1)^*$

---

9) Give regular expressions for the following:

(a) The language of all strings of 0's and 1's that have odd length of 1's.

$(0+11^*)^*1+1(0+11)^*$

(b) The language of all strings of 0's and 1's containing atleast '1'.

$(0+1)^*1+1(0+1)^*$

(c) The language of all strings of 0,1 having length less than or equal to 3.

Sol: length zero $\Rightarrow \epsilon$

length one $\Rightarrow (0+1)$

" two $\Rightarrow (0+1)(0+1)$.

" three $\Rightarrow (0+1)(0+1)(0+1)$.

$\epsilon+(0+1)+(0+1)(0+1)+(0+1)(0+1)(0+1)$

9) Give regular expression to the following language

$L=\{a^{2n}b^{2m+1} \mid n\geq 0, m\geq 0\}$

Sol $(aa)^*(bb)^*b$

(8) Find regular expression over $\{0,1\}^*$ for the language of strings containing odd length.

Sol: r.e for strings of length 1 is $(0+1)$

" " " 3 is $(0+1)(0+1)(0+1)$

Similarly

r.e odd length is $\underline{(0+1)[(0+1)(0+1)]^*}$

(9) write a r.e over the alphabet $\{0,1\}$ to accept set of all strings that begin or end with 00 or 11

Sol: begin with 00 or 11 $\Rightarrow$ $(00+11)(0+1)^*$

end with 00 or 11 $\Rightarrow$ $(0+1)^*(00+11)$

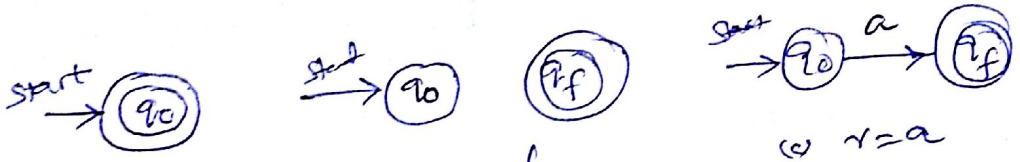
Both : $(00+11)(0+1)^* + (0+1)^*(00+11)$

Finite automata and Regular expressions:

Theorem:
Let r be a regular expression, then there exists an NFA with ϵ-transitions that accept $L(r)$.

Proof: we show by induction on the no of operators in the regular expression r that there is an NFA M with ϵ-transition having one final state and no transitions out of this final state, such that $L(M) = L(r)$.

Basis (Zero operators): The expression r must be ϵ, ϕ (or) a for some a in Σ.



(a) $r = \epsilon$ (b) $r = \phi$ (c) $r = a$

Finite automata for basic step

Induction (One or more operators)

Case 1: $r = r_1 + r_2$. There are NFA's $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$ and $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, \{f_2\})$ with $L(M_1) = L(r_1)$ and $L(M_2) = L(r_2)$

Let q_0 be a new initial state and f_0 a new final state.

construct $M = (Q_1 \cup Q_2 \cup \{q_0, f_0\}, \Sigma_1 \cup \Sigma_2, \delta, q_0, \{f_0\})$.

where δ is defined by

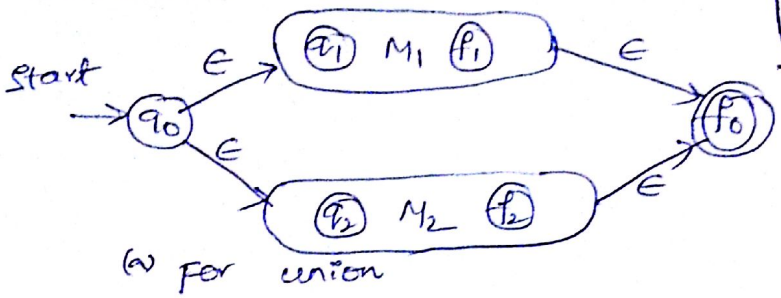
(i) $\delta(q_0, \epsilon) = \{q_1, q_2\}$

(ii) $\delta(q, a) = \delta_1(q, a)$ for q in $Q_1 - \{f_1\}$ and a in $\Sigma_1 \cup \{\epsilon\}$

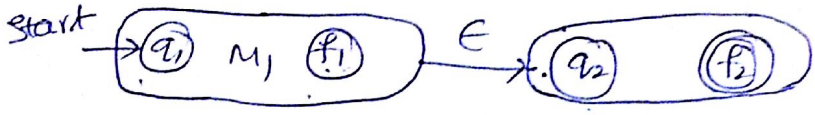
(iii) $\delta(q, a) = \delta_2(q, a)$ " " $Q_2 - \{f_2\}$ and a in $\Sigma_2 \cup \{\epsilon\}$

(iv) $\delta(f_1, \epsilon) = \delta_1(f_2, \epsilon) = \{f_0\}$

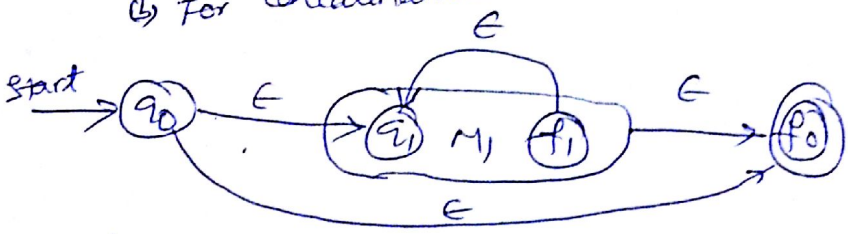
Constructions:



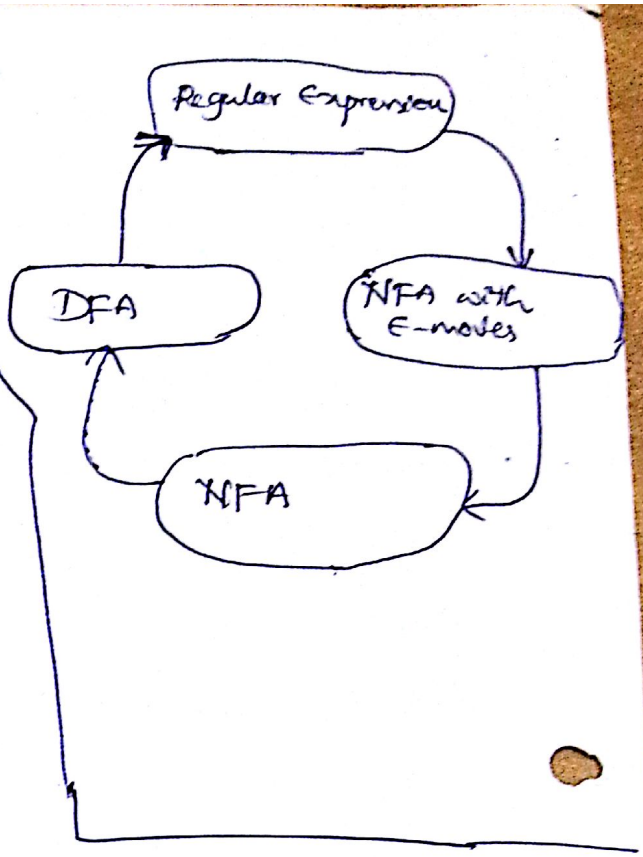
(a) For union



(b) For Concatenation



(c) For closure.



Case 2: $r = r_1 r_2$, Let M_1 and M_2 be as in case 1 and construct $M = (Q_1 \cup Q_2;\ \Sigma_1 \cup \Sigma_2,\ \delta,\ \{q_1\}, \{f_2\})$ where δ is given by

(i) $\delta(q,a) = \delta_1(q,a)$ for q in $Q_1 - \{f_1\}$ and a in $\Sigma_1 \cup \{\epsilon\}$,

(ii) $\delta(f_1, \epsilon) = \{q_2\}$

(iii) $\delta(q,a) = \delta_2(q,a)$ for q in Q_2 and a in $\Sigma_2 \cup \{\epsilon\}$.

Case 3: $r = r_1^*$, let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$, construct $M = (Q_1 \cup \{q_0, f_0\}, \Sigma_1, \delta, q_0, \{f_0\})$, where δ is given by

(i) $\delta(q_0, \epsilon) = \delta(f_1, \epsilon) = \{q_1, f_0\}$

(ii) $\delta(q,a) = \delta_1(q,a)$ for q in $Q_1 - \{f_1\}$ and a in $\Sigma_1 \cup \{\epsilon\}$

→ Thus there is a path in M from q_0 to f_0 labeled x iff we can write $x = x_1 x_2 \cdots x_j$ for some $j > 0$ such that each x_i in $L(M_1)$.

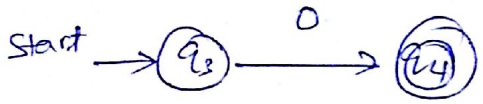
Ex: construct an NFA for the regular expression $01^* + 1$.

Let $r = 01^* + 1$

$r = r_1 + r_2$ where $r_1 = 01^*$ and $r_2 = 1$.

(a) $r_2 = 1$

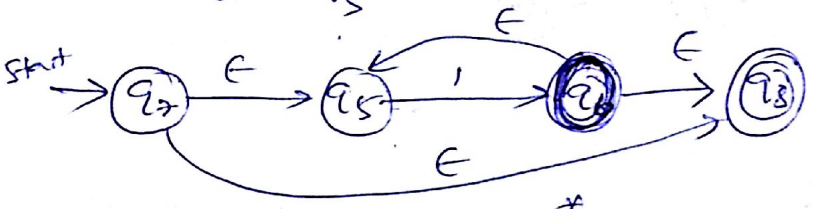
r_1 can be expressed as $r_3 r_4$ where $r_3 = 0$ and $r_4 = 1^*$



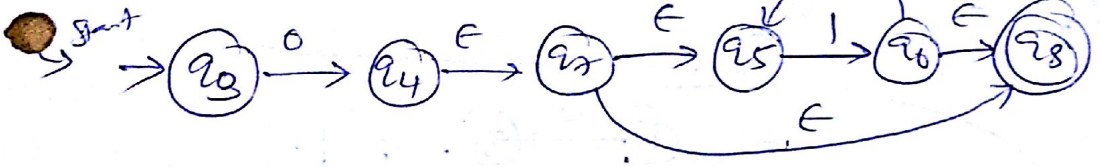
(b) $r_3 = 0$

$r_4 = r_5^*$ where $r_5 = 1$.

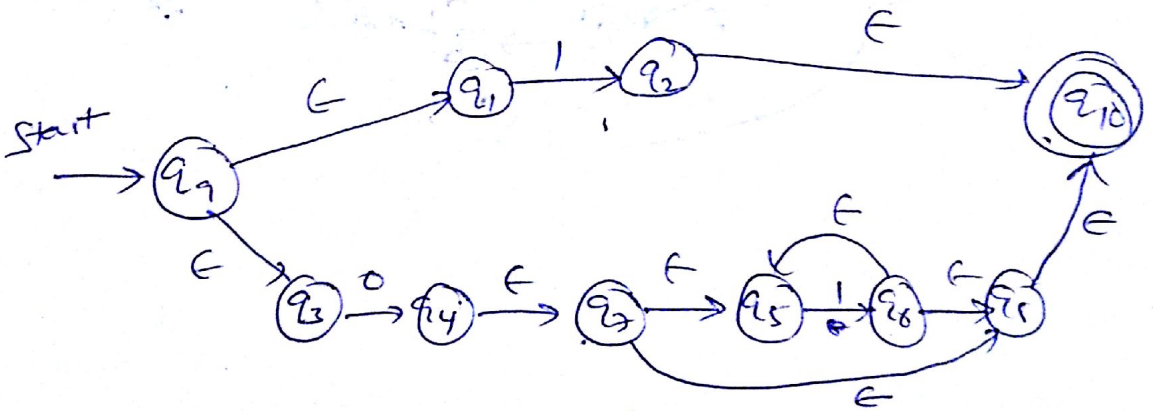
(c) $r_5 = 1$



(d) $r_4 = r_5^* = 1^*$



(e) $r_1 = r_3 r_4 = 01^*$

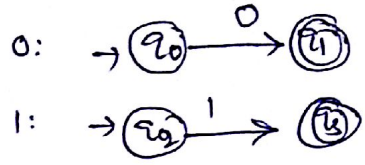


(f) $r = r_1 + r_2 = 01^* + 1$

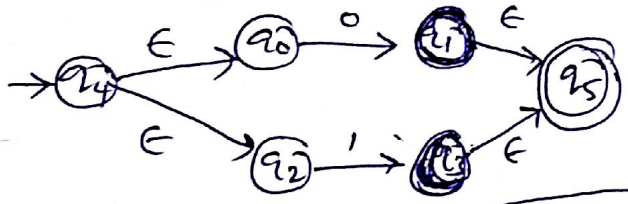
(f) Construct an FA equivalent to the regular expression:

$$(0+1)^*\,(01)\,(0+1)^*$$

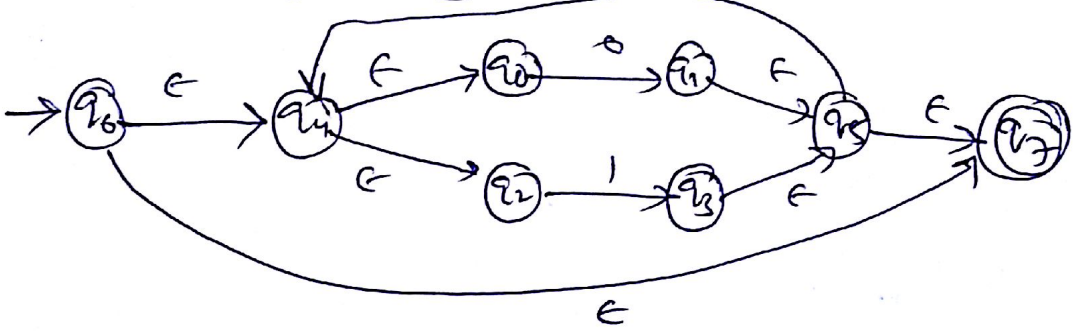
$0+1$:



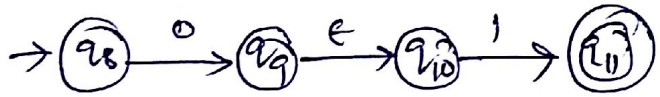
$0+1$

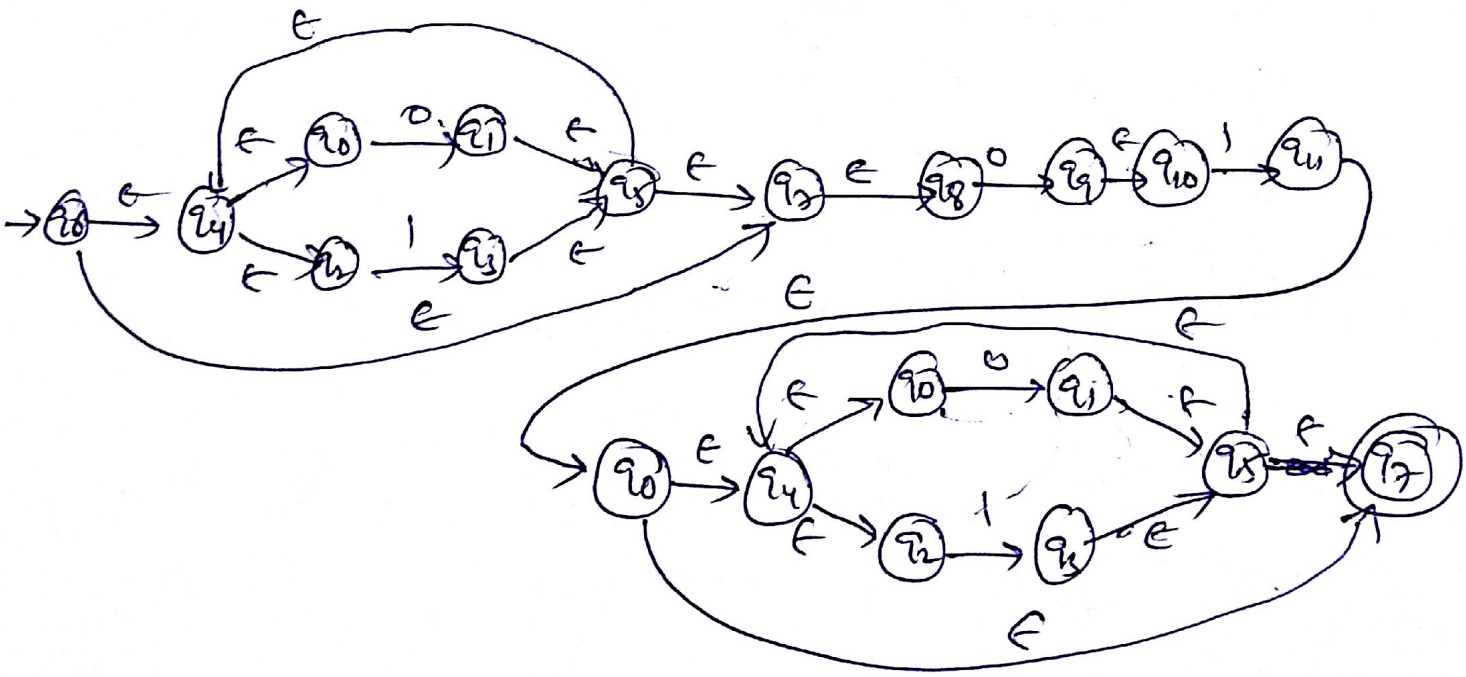


$(0+1)^*$



01 :

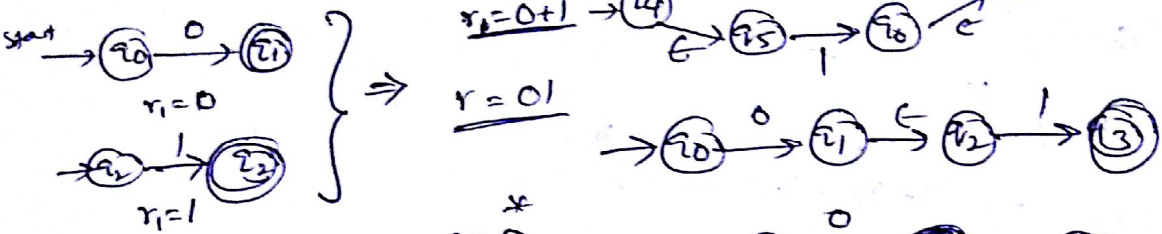


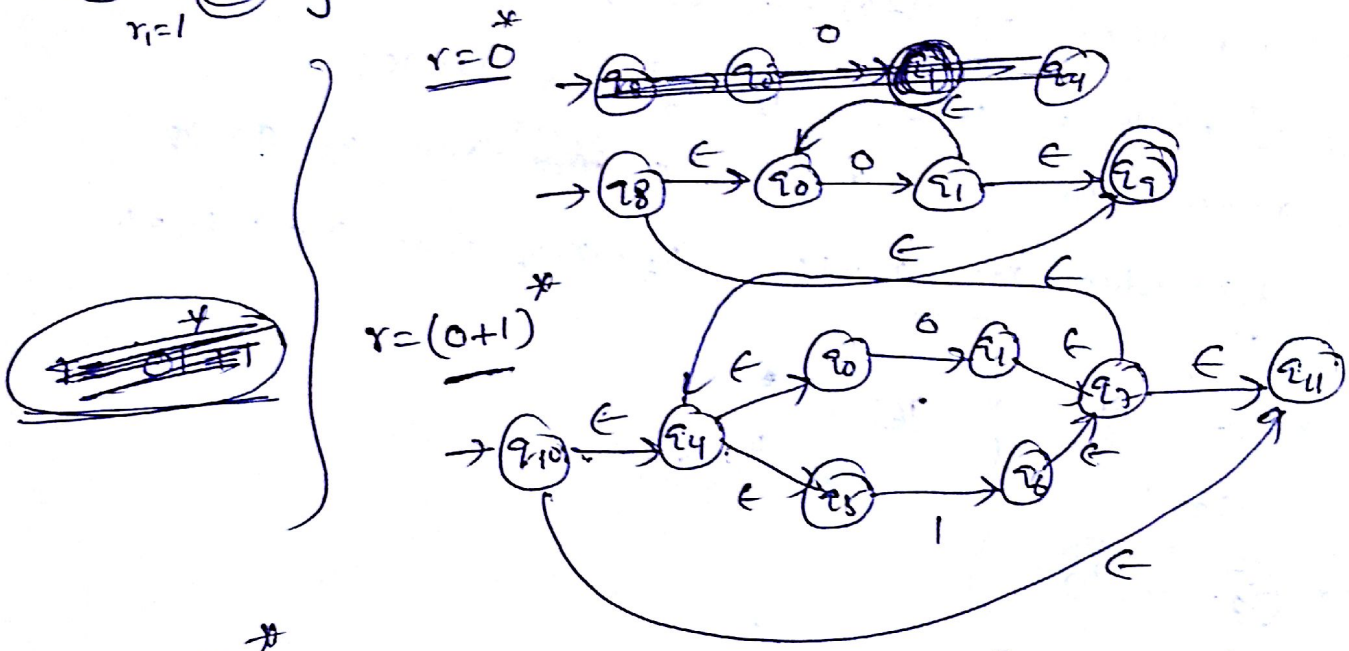


Construct NFA [with ϵ-moves] to the following regular expressions. (3)

$r = 0+1$

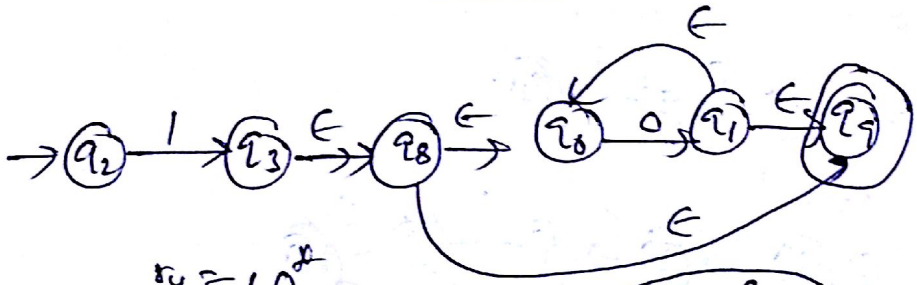
$r = r_1 + r_2$ where $r_1 = 0$, $r_2 = 1$



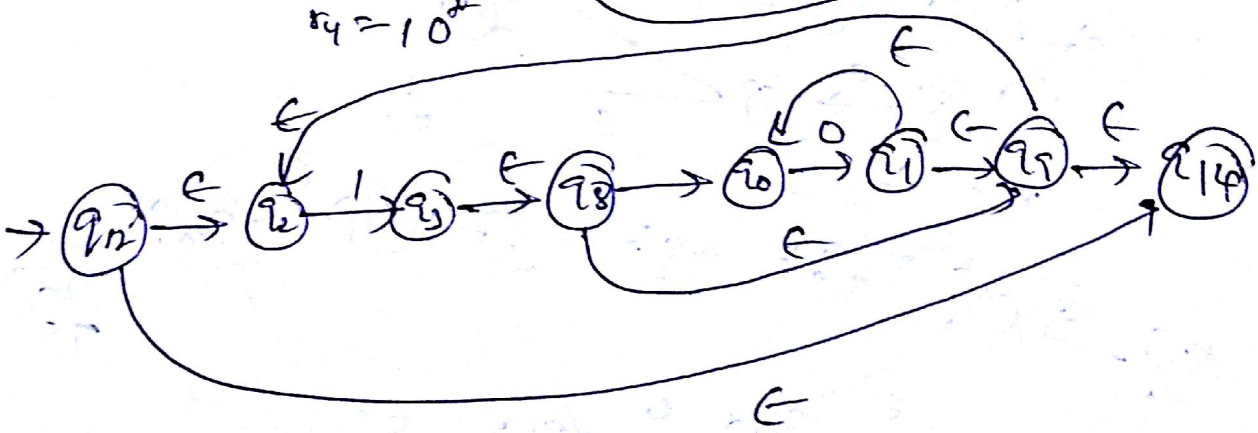


$r = (10^*)^*$

r_4



$r_4 = 10^*$



construct F.A to the following r.e s:

Q) ① $(11+0)^*(00+1)^*$

② $(0+1)^*(11+00)(0+1)^*$

③ $((0+1)(0+1))^*$

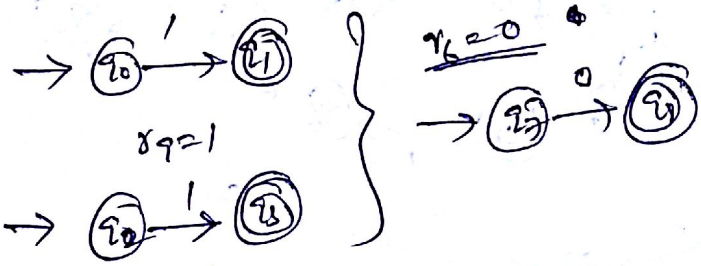
① $r=(11+0)^*(00+1)^*$

$r = r_1 r_2$ where $r_1 = (11+0)^*$, $r_2 = (00+1)^*$

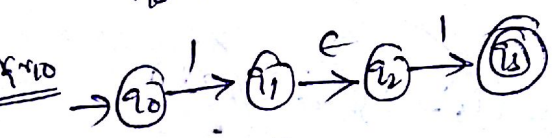
$r_1 = r_3^*$ where $r_3 = 11+0$; $r_2 = r_4^*$ where $r_4 = 00+1$

$r_3 = r_5 + r_6$ where $r_5 = 11$, $r_6 = 0$; $r_4 = r_7 + r_8$ where $r_7 = 00$, $r_8 = 1$

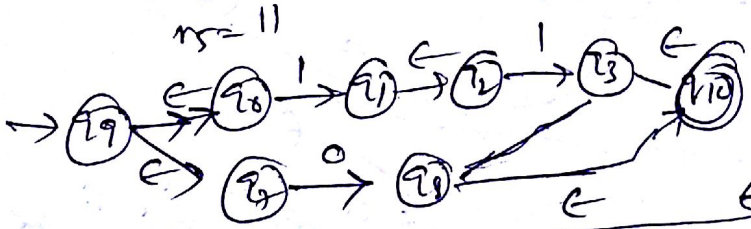
$r_5 = r_9 r_{10}$ where $r_9 = 1$, $r_{10} = 1$; $r_7 = r_{11} r_{12}$ where $r_{11} = 0$, $r_{12} = 0$



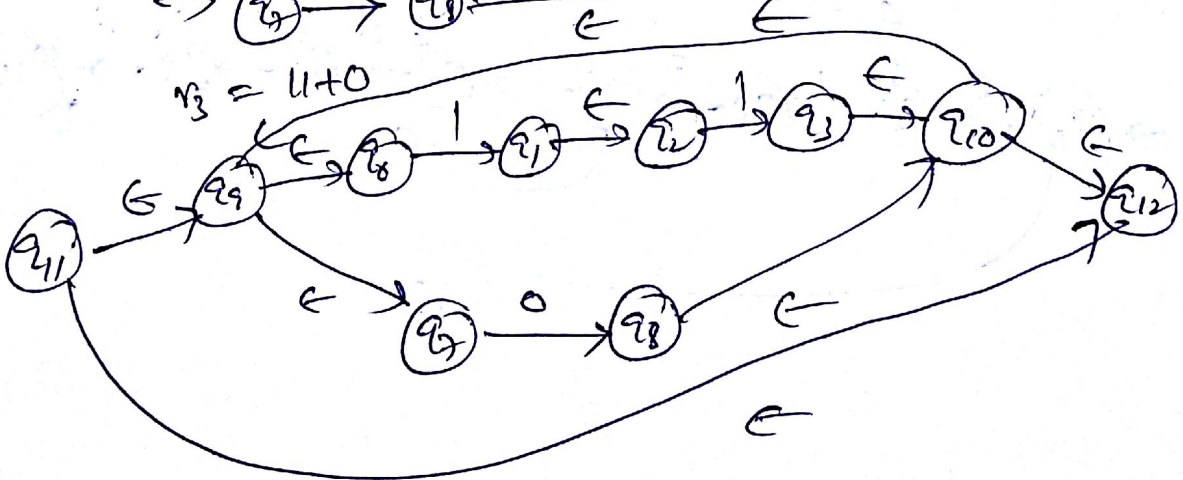
$r_5 = r_9 r_{10}$



$r_5 = 11$



$r_3 = 11+0$



$r_1 = (11+0)^*$

Arden's Theorem:

Let P and Q be two regular expressions over Σ. If P does not contain ϵ, then the following equation in R,

$$\boxed{R = Q + RP} \longrightarrow \text{Eq}(1)$$

has a unique solution given by $\boxed{R = QP^*}$

Proof: $R = Q + RP$

$$Q + (QP^*)P = Q(\epsilon + P^*P) = Q(P^*) = QP^*$$

Hence Eq(1) is satisfied when $R = QP^*$. This means $R = QP^*$ is a solution of Eq(1).

To prove uniqueness, Consider Eq(1). Here, replacing R by $Q + RP$ on the R.H.S., we get the equation

$$\begin{aligned} Q + RP &= Q + (Q + RP)P \\ &= Q + QP + RP^2 \\ &= Q + QP + QP^2 + \cdots + QP^i + RP^{i+1} \\ &= Q(\epsilon + P + P^2 + \cdots + P^i) + RP^{i+1} \end{aligned}$$

$$R = Q(\epsilon + P + P^2 + \cdots + P^i) + RP^{i+1} \quad \text{for } i \geq 0 \longrightarrow \text{Eq}(2)$$

We now show that any solution of Eq(1) is equivalent to QP^*. Suppose R satisfies Eq(1) then it satisfies Eq(2). Then w belongs to the set $Q(\epsilon + P + P^2 + \cdots + P^i) + RP^{i+1}$. As P does not contain ϵ, RP^{i+1} has no string of length less than $i+1$ and so w is not in the set RP^{i+1}. This means w belongs to the set $Q(\epsilon + P + P^2 + \cdots + P^i)$, and hence to QP^*. Thus R and QP^* represent the same set. This proves the uniqueness of the solution of Eq(1).

Conversion of F.A to regular expression:

Arden's Theorem

Let P and Q be two regular expressions over Σ. If P doesnot contain ϵ, then the following equation in R,

$$R = Q + RP$$

has a unique solution given by $R = QP^*$.

Assumptions

① The transition diagram (FA) doesnot have ϵ-moves.

② It has only one initial state, say v_1

③ Its vertices (states) are $v_1, v_2 \ldots v_n$

④ v_i is the r.e corresponding the set of strings accepted by the system even though v_i is a final state.

⑤ α_{ij} denotes the r.e. representing the set of labels of edges from v_i to v_j. when there is no such edge, $\alpha_{ij} = \phi$,

consequently we get the following set of equations in $v_1, v_2 \ldots v_n$

$$\begin{cases} v_1 = v_1\alpha_{11} + v_2\alpha_{21} + \ldots + v_n\alpha_{n1} + \epsilon \\ v_2 = v_1\alpha_{12} + v_2\alpha_{22} + \ldots + v_n\alpha_{n2} \\ \vdots \\ v_n = v_1\alpha_{1n} + v_2\alpha_{2n} + \ldots + v_n\alpha_{nn} \end{cases}$$

By repeatedly applying substitutions and Arden's theorem, we can express v_i in terms of α_{ij}'s.

For getting the set of strings recognised by the transition system, we have to take the union of all v_i's corresponding to final states.

(9) Convert the following F.A into R.E

Sol:

The three equations for q_1, q_2, and q_3 are

$$\begin{cases} q_1 = q_1a + q_2b + \epsilon \longrightarrow (1) \\ q_2 = q_2b + q_3a + q_1a \longrightarrow (2) \\ q_3 = q_2a \longrightarrow (3) \end{cases}$$

Substitute q_3 in q_2-equation (eq (2)), we get

$$q_2 = q_1a + q_2b + q_2aa$$
$$= q_1a + q_2(b+aa)$$

$\{R = Q + RP, \; R = QP^*\}$

$$\boxed{q_2 = q_1a(b+aa)^*}$$ ✓ [By using Arden's theorem]

Substitute q_2 in q_1-equation (eq (1)), we get

$$q_1 = q_1a + q_1a(b+aa)^*b + \epsilon$$
$$= q_1(a + a(b+aa)^*b) + \epsilon$$
$$q_1 = \epsilon + q_1(a + a(b+aa)^*b)$$

$$q_1 = \epsilon(a+a(b+aa)^*b)^*$$ ✓ [By using Arden's theorem]

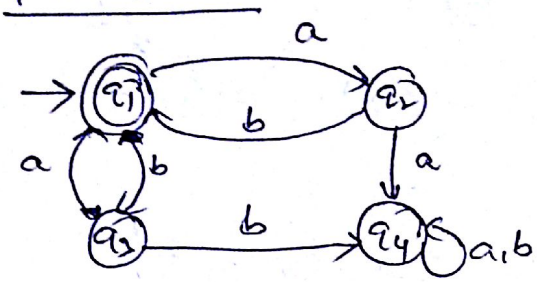
$$q_2 = (a+a(b+aa)^*b)^*a(b+aa)^*$$ ✓

$$\boxed{q_3 = (a+a(b+aa)^*b)^*a(b+aa)^*a}$$

Since q_3 is a final state, the set of strings recognised by the F.A is given by

$$(a+a(b+aa)^*b)^*a(b+aa)^*a$$ ✓

(4). F.A → R.Es



F.A

Sol: Equations

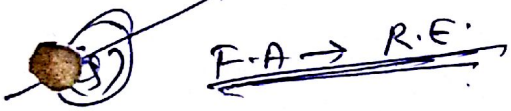
$$q_1 = \epsilon + q_2 b + q_3 b$$
$$q_2 = q_1 a$$
$$q_3 = q_1 b$$
$$q_4 = q_2 a + q_3 b + q_4 a + q_4 b$$

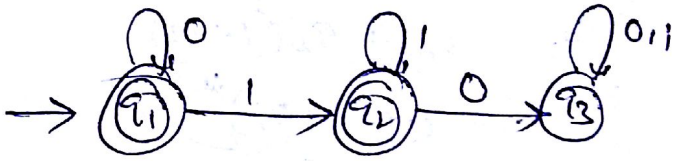
By substituting q_2, q_3 in q_1, we get

$$q_1 = \epsilon + q_1 ab + q_1 ba$$
$$= \epsilon + q_1 (ab + ba)$$
$$= \epsilon (ab + ba)^*$$

$q_1 = (ab+ba)^*$ Regular language is $(ab+ba)^*$

F.A → R.E.





Sol: q_4

$$q_1 = \epsilon + q_1 0 \rightarrow (1)$$
$$q_2 = q_1 1 + q_2 1 \rightarrow (2)$$
$$q_3 = q_2 0 + q_3 0 + q_3 1 \rightarrow (3)$$

By applying arden's theorem

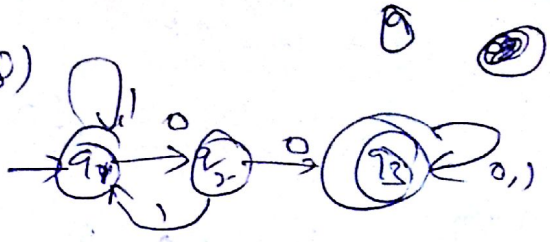
$$q_1 = \epsilon 0^* = 0^*$$
$$q_2 = 0^* 1 + q_2 1 = (0^* 1) 1^*$$

As the final states are q_1 and q_2, we need not solve for q_3

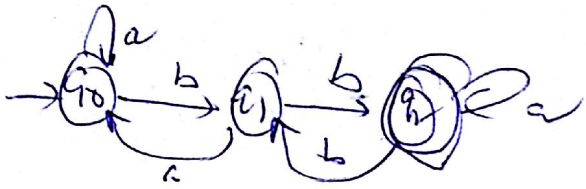
$$q_1 + q_2 = 0^* + 0^* 1 1^* = 0^* (\epsilon + 1 1^*) = 0^* 1^*$$

↑ Regular expression

(5)

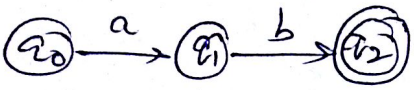


(6)



F.A to R.E

Sol:



Equations:

$q_0 = \epsilon$ *

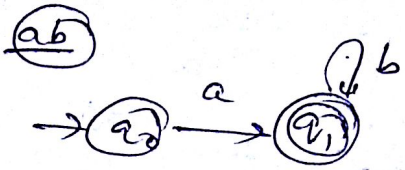
$q_1 = q_0 a$

final state $q_2 = q_1 b$

$q_1 = \epsilon a = a$

$q_2 = ab$ ✓

Sol.



$q_0 = \epsilon$

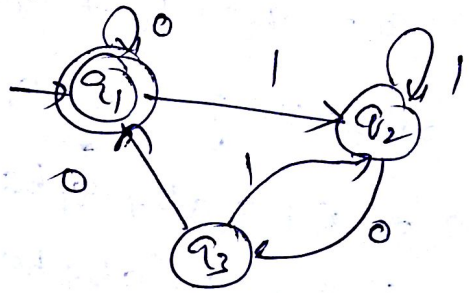
$q_1 = q_0 a + q_1 b$

$q_1 = \epsilon a + q_1 b$

$q_1 = a + q_1 b$

$= ab^*$ [By using Arden's theorem]

Construct a regular expression corresponding to the state diagram given below:



Sol: Equations:

final state $q_1 = \epsilon + q_1 0 + q_3 0$

$q_2 = q_1 1 + q_2 1 + q_3 1$

$q_3 = q_2 0$

By substituting q_3 in q_2

$q_2 = q_1 1 + q_2 1 + q_2 01$

$q_2 = q_1 1 + q_2 (1 + 01)$

$q_2 = q_1 1 (1+01)^*$

$\begin{cases} R = Q + RP \\ R = QP^* \end{cases}$

$q_1 = \epsilon + q_1 0 + q_2 00$

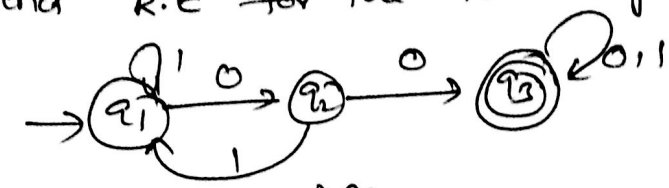
$= \epsilon + q_1 0 + q_1 1 (1+01)^* 00$

$= \epsilon + q_1 (0 + 1(1+01)^* 00)$

$q_1 = \epsilon (0 + 1(1+01)^* 00)^* = (0 + 1(1+01)^* 00)^*$

Regular expression

(P) Find R.E for the following automata



Sol: Equations are

$q_1 = \epsilon + q_1 0 + q_2 1 \rightarrow (1)$

$q_2 = q_1 0 \rightarrow (2)$

$q_3 = q_2 0 + q_3 0 + q_3 1 + \rightarrow (3)$

From eq (3) $q_3 = q_2 0 + q_3 (0+1)$

Arden's theorem $\Rightarrow q_3 = q_2 0 (0+1)^* \rightarrow (4)$

substitute (2) in (1)

$q_1 = \epsilon + q_1 0 + q_1 01$

$= \epsilon + q_1 (0+01)$

$q_1 = (0+01)^* \rightarrow (5)$

substitute (5) in (2)

$q_2 = (0+01)^* 0 \rightarrow (6)$

Substitute (6) in (4)

$q_3 = (0+01)^* 00 (0+1)^*$

R.E is $(0+01)^* 00 (0+1)^*$ ✓

(Q) Show that the following language is not regular [pumping lemma].

$L = \{ a^n b a^n \mid n \geq 0 \}$

Sol: Let $z = a^n b a^n$

Set of strings accepted by the language is

$L = \{ b, aba, aabaa \ldots \}$

Choose a string from the language and apply pumping lemma

$z = \underbrace{a}_{u} \underbrace{b}_{v} \underbrace{a}_{w}$ $\quad |uv| \leq n, |v| \geq 1$

$uv^2w = abba \notin L$

So, L is not regular.

(Q) Find R.E to the following Automata

Sol: Equations are

$A = \epsilon + A0 \rightarrow (1)$

$B = A1 + B1 \rightarrow (2)$

$C = B0 + C0 + C1 \rightarrow (3)$

From (1), using Arden's theorem $[R = Q + RP \Rightarrow R = QP^*]$

$A = \epsilon(0)^* = 0^* \rightarrow (4)$

Substitute (4) in (2)

$B = 0^*1 + B1$

$\Rightarrow B = 0^*1^* \rightarrow (5)$ [using Arden's theorem]

Sub (5) in (3)

$C = 0^*1^*0 + C(0+1)$

$\Rightarrow C = 0^*1^+0(0+1)^*$

Final states are A and B, so R.E is A + B

$A + B = 0^* + 0^*1^* \Rightarrow 0^*(\epsilon + 1^*) = 0^*(\epsilon + 1^+) = \underline{0^*1^*}$

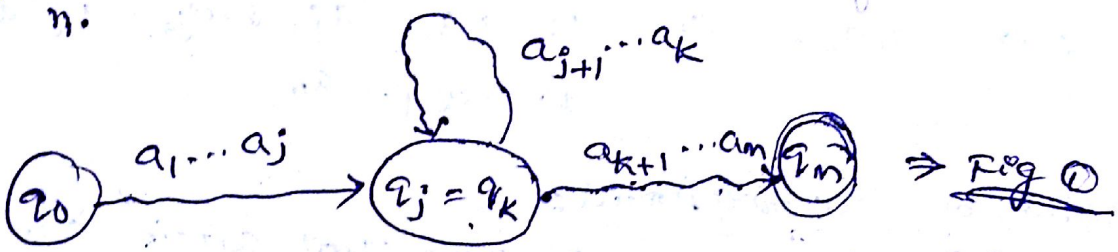
pumping lemma

→ A Language is a regular set (regular) if it is the set accepted by some finite automata.

→ pumping lemma is a powerful tool for proving certain languages ~~not~~ nonregular. It is also useful to check the Language accepted by a given FA is infinite or finite.

If a language is regular, it is accepted by a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with some particular no of states, say n. Consider an input of n or more symbols $a_1, a_2 \cdots a_m$, $m \geq n$, and for $i = 1, 2, \ldots m$, let $\delta(q_0, a_1 a_2 \cdots a_i) = q_i$.

It is not possible for each of the $n+1$ states $q_0, q_1 \cdots q_n$ to be distinct, since there are only n different states. Thus there are two integers j and k, $0 \leq j < k \leq n$, such that $q_j = q_k$. since $j < k$, the string $a_{j+1} \cdots a_k$ is of length atleast 1, and since $k \leq n$, its length is no more than n.



⇒ Fig ①

If q_m is in F, that is, $a_1 a_2 \cdots a_m$ is in L(M), then $a_1 a_2 \cdots a_j a_{k+1} a_{k+2} \cdots a_m$ is also in L(M), since there is a path from q_0 to q_m that goes through q_j but not around the loop labelled $a_{j+1} \cdots a_k$.

$$\delta(q_0, a_1 a_2 \cdots a_j a_{k+1} \cdots a_m) = \delta(\delta(q_0, a_1 \cdots a_j), a_{k+1} \cdots a_m)$$
$$= \delta(q_j, a_{k+1} \cdots a_m)$$
$$= \delta(q_k, a_{k+1} \cdots a_m)$$
$$= q_m$$

similarly, we could go around the loop of fig ① more than once - in fact, as many times as we like.

Thus, $a_1 \cdots a_j (a_{j+1} \cdots a_k)^i a_{k+1} \cdots a_m$ is in $L(M)$ for any $i \geq 0$.

For any sufficiently long string accepted by an FA, we can find a substring near the beginning of the string that may be "pumped", i.e. repeated as many times as we like, and the resulting string will be accepted by the FA.

→ The formal statement of the pumping lemma follows:

Lemma: Let L be a regular set. Then there is a constant n such that if z is any word in L, and $|z| \geq n$, we may write $z = uvw$ in such a way that $|uv| \leq n$, $|v| \geq 1$, and for all $i \geq 0$, uv^iw is in L. Furthermore, n is no greater than the no of states of the smallest FA accepting L.

z is $a_1 a_2 \cdots a_m$, $u = a_1 a_2 \cdots a_j$, $v = a_{j+1} \cdots a_k$ and $w = a_{k+1} \cdots a_m$

Application of pumping lemma:

→ This theorem can be used to prove that certain sets are non regular.

Steps:

① Assume L is a regular. Let n is the no of states in the corresponding FA.

② choose a string z such that $|z| \geq n$; use pumping lemma to write $z = uvw$, with $|uv| \leq n$ and $|v| \geq 1$

③ Find a suitable integer i such that $uv^iw \notin L$. This contradicts our assumption. Hence L is not regular.

→ The decomposition is valid only for strings of length greater than or equal to the no of states.

Show that the following languages are not regular: (non)

① $L = \{a^n b^n \mid n \geq 1\}$

Sol: $L = \{ab, aabb, aaabbb, aaaabbbb \ldots\}$

Assume 'L' is regular.

Take a string $z = aabb$

Divide the above string as $z = uvw$ $\quad \{|uv| \leq n,\ |v| \geq 1\}$

Case(i) $z = \underset{u}{a}\ \underset{v}{a}\ \underset{w}{bb}$

Now, $uv^0w = aa^0bb = aabb \notin L$

$uv^3w = aa^3bb = aaaabb \notin L$

Case(ii) $z = \underset{u}{a}\ \underset{v}{ab}\ \underset{w}{b}$

$uv^0w = a(ab)^0b = aabb \notin L$

$uv^3w = a(ab)^3b = aababab \notin L$

since $uv^iw \notin L$, our assumption is wrong.

So, By contradiction, 'L' is not regular.

② $L = \{ww \mid w \in (a+b)^*\}$ $\quad w \in \{\epsilon, a, b, aa, ab, ba, bb \ldots\}$

L: $L = \{\epsilon, \underset{u}{\underset{w}{aa}}, \underset{u}{\underset{w}{bb}}, \underset{u}{aa}\ \underset{w}{aa}, \underset{u}{bb}\ \underset{w}{bb}, abab, baba \ldots\}$

Assume 'L' is regular

Take $z = \underset{u}{a}\ \underset{v}{aa}\ \underset{w}{aa}$

$uv^0w = aa^0a = aaaa \notin L$

$uv^3w = aaa^3a = aaaaaa \in L$

for $i=2$, $uv^iw \notin L$,

our assumption is wrong.

By contradiction, L is non regular

$z = \underset{u}{a}\ \underset{v}{ba}\ \underset{w}{ba}$

$uv^0w = a(ba)^0ba = ababa \notin$

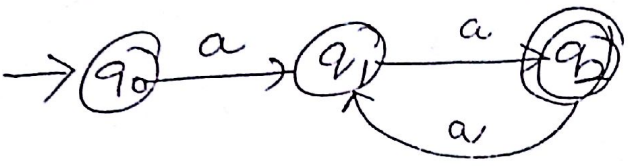
$uv^3w = a(ba)^3ba$

$= ababababa \in L$

16(a) Check the following languages are regular (or) not.

(1) $L = \{ a^{2n} / n \geq 1 \}$ → $n = \{1, 2, 3, 4, \ldots\}$ ($n \geq 1$)

Sol: $L = \{ aa, aaaa, aaaaaa \ldots \}$



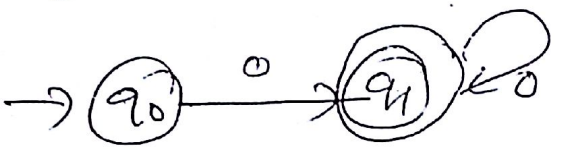
State	Input a
→ q0	q1
q1	q2
(q2)	q1

So, there is a F.A to accept the given language.

The above language $L = \{a^{2n} / n \geq 1\}$ is regular.

(2) $L = \{ 0^n / n \geq 1 \}$

Sol: $L = \{ 0, 00, 000, 0000 \ldots \}$



So, L is regular

Non regular

$L = \{ a^{2n} b^n / n \geq 1 \}$

$L = \{ a^n b^{2n} / n \geq 1 \}$

$L = \{ a^n b a^n / n \geq 1 \}$

$L = \{ w w^R / w \in (0+1)^* \}$

$L = \{ a^n b^n c^n / n \geq 1 \}$

$L = \{ a^p b^p / p \text{ is prime} \}$

$L = \{ 0^{i^2} / i \geq 0 \}$

$L = \{ a^p / p \text{ is prime} \}$

$L = \{ a^n b^n / n \geq 1 \}$

$L = \{ ww / w \in (0+1)^* \}$

Regular

$L = \{ a^{2n} / n \geq 1 \}$

$L = \{ 0^n / n \geq 1 \}$

$L = \{ a^n b / n \geq 0 \}$

$L = \{ b a^n / n \geq 0 \}$

$L = \{ a b^n / n \geq 0 \}$

$L = \{ b^n a / n \geq 0 \}$

Closure properties of Regular sets:

The regular sets are closed under

1. union, concatenation and Kleen closure.
2. complementation.
3. Intersection
4. ~~Substitution~~ reverse
5. Difference
6. ~~Homomorphism~~ / Inverse Homomorphism

union: $r+s$

rs, r^*, s^*

$r_1 \cap r_2$, $r \cap s$

$r - s = r \cap s'$

$r' = \Sigma^* - r$

$L \rightarrow L^R$

① The regular sets are closed under union, concatenation and Kleen closure:

proof:- If L_1 and L_2 are regular languages, there are regular expressions r_1 and r_2 that define ~~the~~ two languages. Then $(r_1 + r_2)$ is a regular expression that defines the language $L_1 + L_2$. The language $L_1 L_2$ can be defined by regular expression $r_1 r_2$. The language L_1^* can be defined by the regular expression $(r_1)^*$. Therefore all three sets of words are definable by regular expressions and so are themselves regular languages.

② The class of regular sets is closed under complementation:

Proof: To show closure under complementation

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

$M' = (Q, \Sigma, \delta, q_0, Q - F)$ accepts $\bar{L}_1$.

Assume δ^* to be a total function, so that $\delta^*(q_0, w)$ is defined for all $w \in \Sigma^*$, consequently either $\delta^*(q_0, w)$ is a final state, in which case $w \in L_1$ or $\delta^*(q_0, w) \in Q - F$ and $w \in \bar{L}$

③ The regular sets are closed under intersection.

Proof: Intersection operation can be performed by closure under union and complementation.

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

The intersection of two regular sets can be constructed taking cartesian product of states and we sketch the construction as follows.

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F)$ and

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be two DFA.

Let $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$

where for all P_1 in Q_1, P_2 in Q_2 and a in Σ,

$$\delta = ([P_1, P_2], a) = [\delta_1(P_1, a), \delta(P_2, a)]$$

④ ~~show~~ The regular sets are closed under difference.

Proof: The set difference can be written as

$$L_1 - L_2 = L_1 \cup \overline{L_2}$$

If L_2 is regular then $\overline{L_2}$ is also regular then we have already proved that regular languages are closed under intersection.

so $L_1 \cup \overline{L_2}$ is also regular.

→ If L is a regular language, and h is a homomorphism on its alphabet, then $h(L) = \{ h(w) \mid w \text{ is in } L \}$ is also regular.

→ Let h be a homomorphism and 'L' a language whose alphabet is the output language of h

$$h^{-1}(L) = \{ w \mid h(w) \text{ is in } L \}$$ → regular

Q) Construct F.A which accepts the regular expression

(i) 0^*1+10 (ii) 01^*+10^*

Sol:

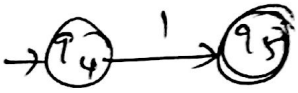
(i) Given R.E is $R = 0^*1+10$

Let $R = R_1 + R_2$ where $R_1 = 0^*1$ and $R_2 = 10$

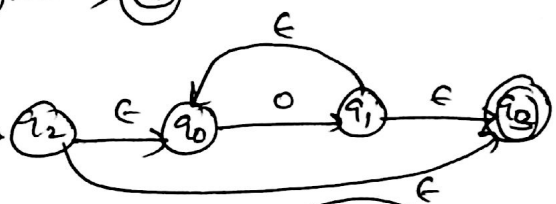
$R_1 = R_3 R_4$ where $R_3 = 0^*$ and $R_4 = 1$

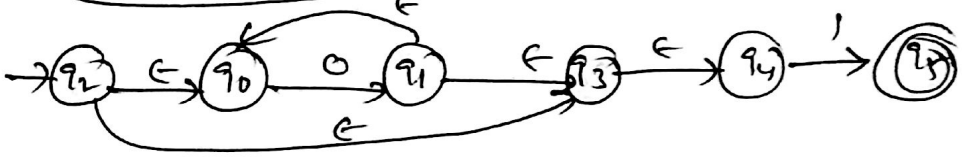
$R_3 = R_5^*$ where $R_5 = 0$

$R_2 = R_6 R_7$ where $R_6 = 1$ and $R_7 = 0$

$R_4 = 1$ 

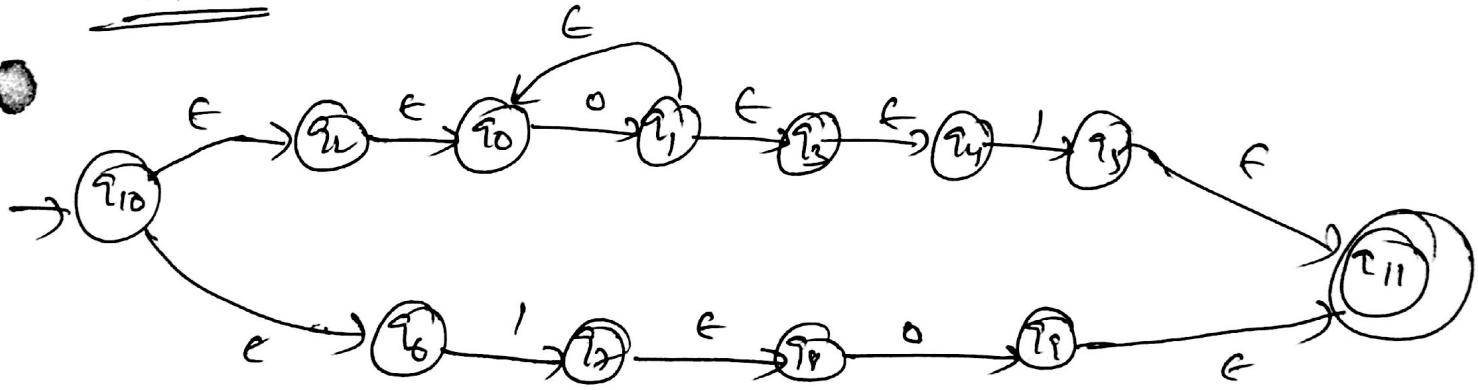
$R_5 = 0$: 

$R_3 = R_5^* = 0^*$ = 

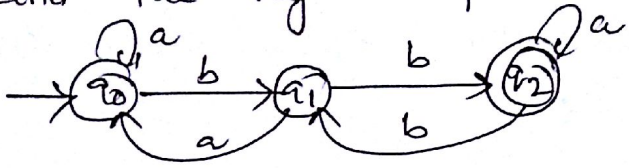
$R_1 = R_3 R_4 = 0^*1$ = 

$R_2 = R_6 R_7 = 10$ 

$R = 0^*1+10$



(Q) Find the regular expression accepted by following FA.



Sol: equations are

$q_0 = \epsilon + q_0a + q_1a \rightarrow (1)$

$q_1 = q_0b + q_2b \rightarrow (2)$

$q_2 = q_1b + q_2a \rightarrow (3)$

Arden's theorem

$R = QP + RP$

$\Rightarrow R = QP^*$

From eq (3), using Arden's theorem

$q_2 = q_1ba^* \rightarrow (4)$

substitute (4) in (2)

$q_1 = q_0b + q_1ba^*b$

$q_1 = q_0b(ba^*b)^* \rightarrow (5)$

substitute (5) in (1)

$q_0 = \epsilon + q_0a + q_0b(ba^*b)^*$

$= \epsilon + q_0(a + b(ba^*b)^*)$

$q_0 = (a + b(ba^*b)^*)^* \rightarrow (6)$

$q_1 = (a + b(ba^*b)^*b)^*(ba^*b)^*$

$q_2 = (a + b(ba^*b)^*b(ba^*b)ba^*$

(Q) Construct a DFA for the regular expression

$r = (a+b)^*abb$

$L = \{abb, aabb, bbabb, ababb, \ldots\}$

DFA q_3

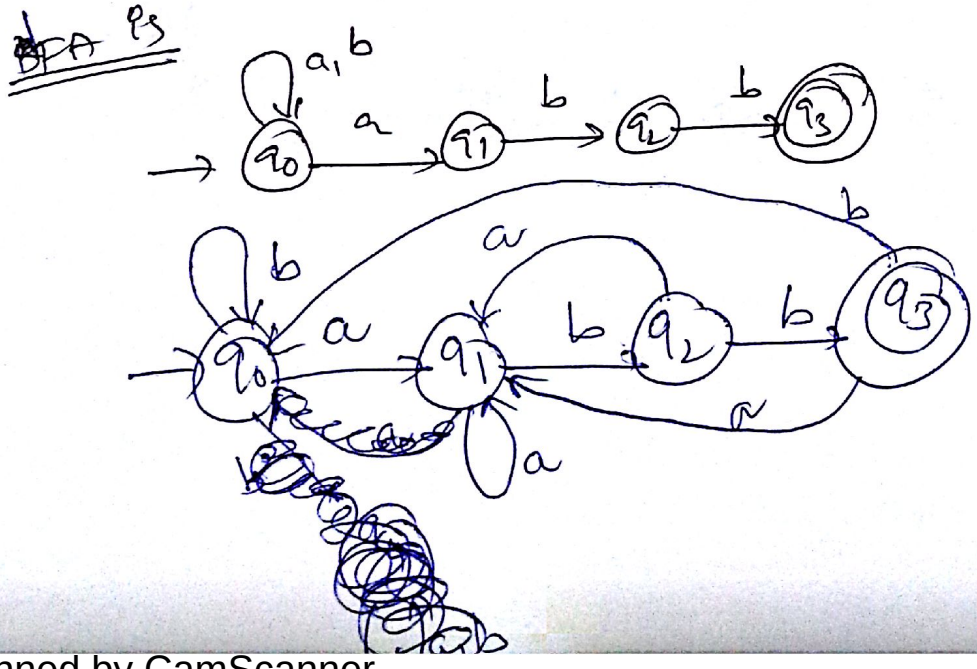


abb abb

abbbb abb

CONTEXT-FREE GRAMMAR:

Ambiguity in CFGs:-

A CFG G such that some word has two derivation trees is said to be ambiguous. If some word has more than one left-most derivation (or) more than one rightmost derivation.

A CFL for which every CFG is ambiguous is said to be an inherently ambiguous CFL.

⇒ A terminal string (word) $w \in L(G)$ is ambiguous if there exists two (or) more derivation trees for w.

Example:

consider $G=(\{S\}, \{a,b,+,*\}, P, S)$ where P consists of

$S \to S+S \mid S*S \mid a \mid b$

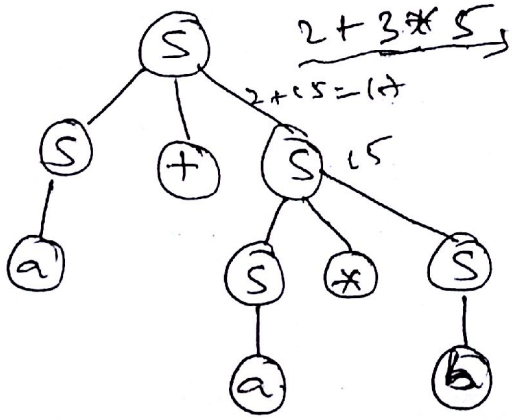
we have two derivation trees for $a+a*b$ given below:

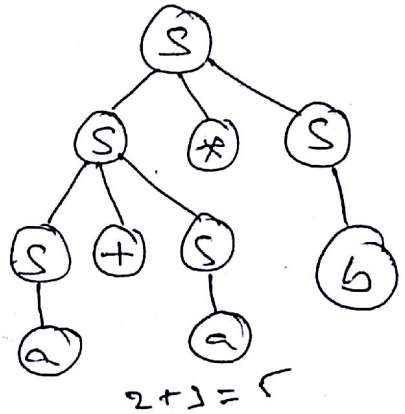
Left-most derivations

① $S \Rightarrow \underline{S}+S$
$\Rightarrow a+\underline{S}$
$\Rightarrow a+\underline{S}*S$
$\Rightarrow a+a*\underline{S}$
$\Rightarrow a+a*b$

② $S \Rightarrow \underline{S}*S$
$\Rightarrow S+\underline{S}*S$
$\Rightarrow a+S*S$
$\Rightarrow a+a*S$
$\Rightarrow a+a*b$

$S \to SAS$,
$A \to a \mid b$





Def:
⇒ A context-free grammar G is ambiguous if there exists some $w \in L(G)$, which is ambiguous.

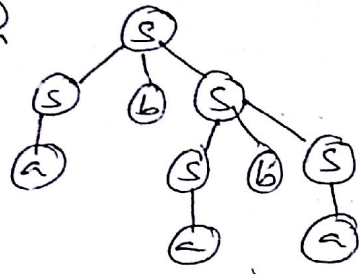
(Q) If G is the grammar $S \to SbS \mid a$, show that G is ambi...

Sol: Given $S \to SbS \mid a$

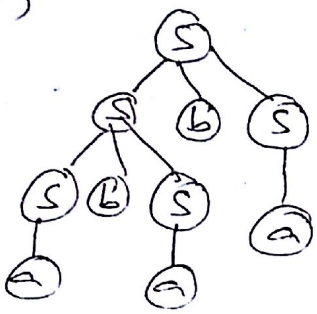
. consider $w = ababa \in L(G)$

$S \Rightarrow SbS$
$\Rightarrow abS$
$\Rightarrow abSbS$
$\Rightarrow ababa$

① derivation tree



②



Thus there exists $w = ababa \in L(G)$ has two derivation trees.

So CFG is ambiguous.

(Q) show that the grammar $S \to a \mid absb \mid aAb$, $A \to bs \mid aAAb$ is ambi...

Sol: Given CFG is

$S \to a \mid abSb \mid aAb$
$A \to bS \mid aAAb$

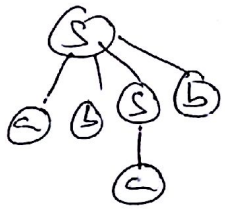
① $S \Rightarrow absb$
$\Rightarrow abab$

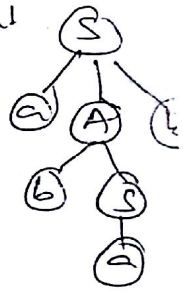
② $S \Rightarrow aAb$
$\Rightarrow absb$
$\Rightarrow abSb$
$\Rightarrow abab$

Derivations:

①
$S \Rightarrow absb$
$S \Rightarrow ababSbb$
$\Rightarrow ababab$

②
$S \Rightarrow absb$
$\Rightarrow abaAb$
$\Rightarrow ababSb$
$\Rightarrow ababab$





So, G is ambiguous.

(Q) S.T the grammar $S \to aB \mid ab$, $A \to aAB \mid a$, $B \to ABb \mid b$ is ambiguous

Sol:

①
$S \Rightarrow aB$
$\Rightarrow aABb$
$\Rightarrow aabb$
$\Rightarrow aaABbb$
$\Rightarrow aaabbb$

②
$S \Rightarrow aB$
$\Rightarrow aABb$
$\Rightarrow aaABBb$
$\Rightarrow aaaBBb$
$\Rightarrow aaabBb$
$\Rightarrow aaabbb$

(Q) Test whether the grammar is ambiguous or not.

① $S \to SS \mid a \mid b$
② $S \to a \mid Sa \mid SSb \mid SbS$
③ $S \to aSb \mid aaSb \mid \epsilon$
④ $S \to 0S1 \mid SS \mid \epsilon$

} ambiguous grammars

⑤ $S \to A \mid B$
$A \to aAb \mid ab$
$B \to abB \mid \epsilon$

} ambiguous

Simplification of context-free grammars:-

If L is a non-empty context-free language then it can be generated by a context-free grammar G with the following properties

(i) Each variable and each terminal of G appears in the derivation of some word in L.

(ii) There is no productions of the form $A \to B$ where A and B are variables.

In a CFG G, it may not be necessary to use all the symbols in $V \cup \Sigma$, or all the productions in P for deriving sentences (words). Here, we try to eliminate symbols and productions in G which are not useful for derivation of sentences (words)

Example: $G = (\{S, A, B, C, E\}, \{a, b, c\}, P, S)$ where P is,

$S \to AB, A \to a, B \to b, B \to C, E \to c/\epsilon$

The language generated by the grammar is $L(G) = \{ab\}$.

$G' = (\{S, A, B\}, \{a, b\}, P', S)$ where P' is

$S \to AB, A \to a, B \to b$ will generate $L(G) = \{ab\}$.

eliminations

(i) C does not derive any terminal string

(ii) E and C do not appear in any sentential form.

(iii) $E \to \epsilon$ is a null production

(iv) $B \to C$ simply replaces B by C.

Theorem 1: If G is a CFG such that $L(G) \neq \phi$, we can find an equivalent grammar G' such that each variable in G' derives some terminal string.

Theorem 2: For every CFG $G = (V, T, P, S)$ we can construct an equivalent grammar $G' = (V', T', P', S)$ such that every symbol in $V' \cup T'$ appears in some sentential form

Dependency graph: A dependency graph has its vertices labelled with variables, with an edge b/w vertices A and B iff if there is production of the form

$A \to xBy$

Elimination of useless symbols:-

Let $G = (V, T, P, S)$ be a grammar.

→ A symbol X is useful if there is a derivation $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$ for some α, β, and w, where w is in T^*. otherwise X is useless.

Two aspects to usefulness:

→ some terminal string must be derivable from 'X'

→ 'X' must occur in some string derivable from S.

[The symbol can't derived from goal symbol those are useless symbols].

→ for every nonempty CFL is generated by a CFG with no useless symbols.

Example:

Consider the grammar $S \to AB \mid a,\ A \to a$

Applying theorem1, we find that no terminal string is derivable from B, so we eliminate B and the production $S \to AB$

$S \to a$
$A \to a$ (S) (A)

Applying theorem 2, we find that only S and 'a' appear in sentential form. So we eliminate $A \to a$.

Thus $(\{S\}, \{a\}, \{S \to a\}, S)$ is an equivalent grammar with no useless symbols.

Eliminate useless symbols from the grammar

$S \to aS \mid A \mid C,\ A \to a,\ B \to aa,\ C \to acb$

Sol: 'C' doesnot derive any string (terminal), so 'C' is eliminated

$S \to aS \mid A,\ A \to a,\ B \to aa$; (S) (A) (B)

B is also useless symbol [B is not in any sentential form]

so we can eliminate 'B' and its productions

$S \to aS \mid A$
$A \to a$ — this is the required grammar with no useless symbols.

Elimination of ϵ-productions :-

A CFG may have productions of the form $A \to \epsilon$. The production $A \to \epsilon$ is just used to erase. So a production of the form $A \to \epsilon$, where A is a variable, is called a null production.

Def: A variable A in a CFG is nullable if $A \overset{*}{\Rightarrow} \epsilon$.

Ex: Consider G, $S \to aS / bA / \epsilon$

$A \to \epsilon$

we can eliminate $S \to \epsilon$, $A \to \epsilon$ then

$S \to aS / bA$

A can be replaced by ϵ by using $A \to \epsilon$

$S \to aS / b / bA$ grammar without null productions.

Eliminate null productions (ϵ-productions) from the CFG given b

$S \to aS / bA$

$A \to aA / \epsilon$

Sol: $S \to aS$, $S \to bA$ gives $S \to bA$ and $S \to b$

$A \to aA$ gives $A \to aA$ and $A \to a$

$S \to aS / bA / b$

$A \to aA / a$ } required grammar without ϵ-productions.

From the Grammar G, $S \to AaB / aaB$

$A \to \epsilon$

$B \to bbA / \epsilon$ } from this gramma

eliminate ϵ-productions and useless symbols:

Sol: $S \to AaB / aaB$

$A \to \epsilon$

$B \to bbA / \epsilon$

Ex: Example 1

$S \to AaB$ gives $S \to AaB \mid aB \mid Aa \mid a$

$S \to aaB$ gives $S \to aaB \mid aa$

$B \to bbA$ gives $B \to bbA \mid bb$

So, Grammar without null productions is

$$\begin{cases} S \to AaB \mid aB \mid Aa \mid a \\ S \to aaB \mid aa \\ B \to bbA \mid bb \end{cases}$$

2) Eliminate null-productions from the following grammar

$G = (\{S, A, B, D\}, \{a, b\}, \{S \to aS \mid AB,\ A \to \epsilon,\ B \to \epsilon,\ D \to b\}, S)$

Sol: $\begin{cases} S \to aS \text{ gives } S \to aS \text{ and } S \to a \quad [\because S \to AB \text{ gives } S \to \epsilon] \\ S \to AB \text{ gives } S \to B, S \to A, \text{ and } S \to B \end{cases}$

Finally CFG without null productions is

$S \to aS \mid a \mid AB \mid A \mid B$

$D \to b$

3) $\begin{cases} S \to AB \mid \epsilon \\ A \to aASb \mid a \\ B \to bS \end{cases}$

$\begin{cases} S \to AB \\ A \to aASb \text{ gives } A \to aASb \mid aAb \\ B \to bS \text{ gives } B \to bS \mid b \end{cases}$

RG without null productions

$\begin{cases} S \to AB \\ A \to aASb \mid aAb \mid a \\ B \to bS \mid b \end{cases}$

Elimination of unit productions :-

A CFG may have productions of the form $A \to B$, $A, B \in V$.

⇒ A unit production (or chain rule) in a CFG G is a production of the form $A \to B$, where A and B are variables in G

If G is a CFG, we can find a CFG G_1 which has no unit productions such that $L(G_1) = L(G)$.

To remove unit productions from the CFG, the substitution method is considered.

Ex: Consider G with no ϵ-productions,

$S \to A$, $A \to B$, $B \to C$, $C \to D$

Here S, A, B, C are unit variables of length one. The resultant grammar is $S \to D$ [chain rule]

Ex: From the CFG given below, eliminate unit productions

$S \to A | bb$, $A \to B | b$, $B \to S | a$

Sol: unit productions are $S \to A$, $A \to B$, and $B \to S$.

$S \to A$ gives $S \to b$ and $S \to B$ gives $S \to a$ and $S \to bb$

$A \to B$ gives $A \to a$ and $A \to S$ gives $A \to bb$ and $A \to$

$B \to S$ gives $B \to bb$ and $B \to A$ gives $B \to b$, $B \to a$

The new productions are

$$\left.\begin{array}{l} S \to bb | b | a \\ A \to a | bb | b \\ B \to a | b | bb \end{array}\right\}$$ CFG without unit productions

By eliminating useless symbols

$S \to bb | b | a$

(8) Given the grammar $S \to AB$, $A \to a$, $B \to C/b$, $C \to D$, $D \to E$, $E \to a$, Find an equivalent grammar which is reduced and has no unit productions.

Sol: Given grammar is $S \to AB$, $A \to a$, $B \to C/b$, $C \to D$, $D \to E$, $E \to a$

unit productions are $B \to C$, $C \to D$, $D \to E$

① $B \to C$ gives $B \to D$ gives $B \to E$ gives $B \to a$ ✓

② $C \to D$ gives $C \to E$ gives $C \to a$ ✓

③ $D \to E$ gives $D \to a$ ✓

So, Grammar without unit productions is

$S \to AB$, $A \to a$, $B \to a/b$, $C \to a$, $D \to a$, $E \to a$

(9) Remove all unit productions from

$S \to Aa/B$, $B \to A/bb$, $A \to a/bc/B$

Sol:

unit productions are $S \to B$, $B \to A$ and $A \to B$

By using chain rule $S \to A$ without unit productions

① $S \to B$ gives $S \to bb$

$S \to B \to A$ gives $S \to a$, $S \to bc$

② $B \to A$ gives $B \to a$, $B \to bc$

$(B \to A \to B$ gives $B \to bb)$ [optional]

③ $A \to B$ gives $A \to bb$

$(A \to B \to A$ gives $A \to a$, $A \to bc)$ [optional]

$\Rightarrow$

$S \to Aa/bb/a/bc$

$B \to bb/a/bc$

$A \to a/bc/bb$

(10) Remove all unit productions from the following grammar

$S \to A/B/C$, $A \to aAa/B$, $B \to bB/bb$, $C \to aCaa/D$, $D \to baD/abD/aa$

unit productions $S \to A/B/C$, $A \to B$, $C \to D$

~~$S \to A$ gives $S \to aAa$~~

$S \to A \to B$ gives $S \to bb$

$S \to B$ gives $S \to bb$

$S \to C \to D$ gives $S \to aa$

$A \to B$ gives $A \to bb$

$C \to D$ gives $C \to aa$

CFG without unit productions

$S \to bb/aa/AA$, $A \to aAa/bb$, $B \to bB/bb$, $C \to aCaa/aa$, $D \to baD/abD/aa$

Q) Eliminate useless symbols from the given grammar,

$S \to AB \mid CA$, $B \to BC \mid AB$, $A \to a$, $C \to aB \mid b$

Sol: we can't derive a terminal string from B, so eliminate 'B' and its productions

$S \to CA$
$A \to a$
$C \to b$

Grammar without useless symbols.

dependency graph

Q) Eliminate useless symbols

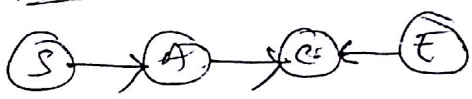
$S \to aAa$
$A \to Sb \mid bCE \mid DaA$
$C \to abb \mid DD$
$E \to aC$, $D \to aDA$

dependency graph

Sol: D doesn't derive any terminal string

$S \to aAa$
$A \to Sb \mid bCE$
$C \to abb$
X ~~$E \to aC$~~ X [doesn't appear in sentential form]

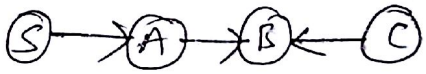
dependency graph



Q) Eliminate useless symbols

$S \to aAa$, $A \to bBB$, $B \to ab$, $C \to aB$

dependency graph



Sol: S, A, B, C derives terminal strings

'C' doesn't appear in any sentential form.

$S \to aAa$, $A \to bBB$, $B \to ab$ [without useless symbols]

(Q) Eliminate useless symbols

$S \to a \mid aA \mid B \mid \epsilon$, $A \to aB \mid \epsilon$, $B \to Aa$, $C \to cCd$, $D \to ddd$

dependency graph

Sol: 'C' is eliminated

$S \to a \mid aA \mid B$, $A \to aB \mid \epsilon$, $B \to Aa$, $D \to ddd$

so 'D' doesn't appear in any sentential form

$S \to a \mid aA \mid B$, $A \to aB \mid \epsilon$, $B \to Aa$

(9) Eliminate all unit, and ϵ-productions from

$S \to AaB \mid aaB$

$A \to D$

$B \to bbA \mid \epsilon$

$D \to E$

$E \to F$

$F \to aS$

Sol: **Elimination of ϵ-productions:**

$B \to \epsilon$ (null production)

$S \to AaB$ gives $S \to AaB \mid Aa$

$S \to aaB$ gives $S \to aaB \mid aa$

$A \to D$

$B \to bbA$

$D \to E, E \to F, F \to aS$

$\Rightarrow$ productions without ϵ-productions

$S \to AaB \mid Aa \mid aaB \mid aa$

$A \to D, B \to bbA$

$D \to E \mid E \to F \mid F \to aS$

Elimination of unit productions:

The unit productions are

$A \to D, D \to E, E \to F$

$A \to D$ gives $A \to aS$ [$\because A \to D \to E \to F \to aS$] (chain rule)

$D \to E$ gives $D \to aS$

$E \to F$ gives $E \to aS$

Thus the grammar after removing unit productions is

$S \to AaB \mid Aa \mid aaB \mid aa$

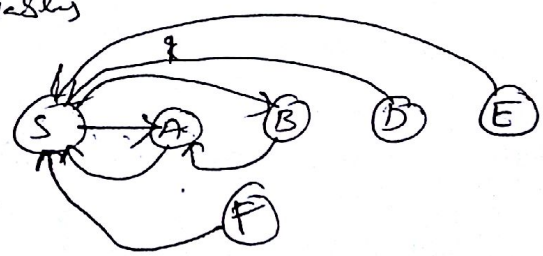
$B \to bbA$

$A \to aS, E \to aS, D \to aS, F \to aS$

Elimination of useless symbols

We can get terminal string from all the variables

E and D are useless symbol as they didn't appear in any sentential form



∴ The optimized grammar is

$S \to AaB \mid Aa \mid aaB \mid aa$

$B \to bbA$

$A \to aS$

pumping lemma for context-free languages:-

⇒ The pumping lemma for regular sets states that every sufficiently long string in a regular set contains a short substring that can be pumped. That is, inserting as many copies of the substring as we like always yields a string in the regular set. The pumping lemma for CFL's states that there are always two short substrings close together that can be repeated, both the same no of times, as often as we like.

⇒ The formal statement of the pumping lemma is as follows:

Lemma: Let L be any CFL. Then there is a constant n, depending only on L, such that if z is in L and $|z| \geq n$, then we may write $z = uvwxy$ such that

① $|vx| \geq 1$

② $|vwx| \leq n$ and

③ for all $i \geq 0$, uv^iwx^iy is in L.

Example: Consider the grammar whose productions are

$S \to AB$, $A \to aB/a$, $B \to bA/b$

$S \Rightarrow AB \Rightarrow aBB \Rightarrow abAB \Rightarrow ababA \Rightarrow ababa$

$\therefore z = ababa$

Q) S.T the following language is not a CFL

Q): $L = \{ a^n b^n c^n \mid n \geq 1 \}$

Sol: $L = \{ abc, aabbcc, aaabbbccc \cdots \}$

Assume 'L' is a context free language

Take $z = \underbrace{aa}_{u}\underbrace{bb}_{v}\underbrace{cc}_{w}\underbrace{}_{x}\underbrace{}_{y}$

Now, $uv^2wx^2y = a(a)^2bb(c)^2c$

$= aaabbccc \notin L$

$uv^3wx^3y = a(a)^3bb(c)^3c$

$= aaaabbcccc \notin L$

for $i \geq r$, $uv^iwx^iy \notin L$

So, our assumption is wrong.

By contradiction, L is not a context free language.

8) S.T the following language is not a CFL

$L = \{ ww \mid w \in (0+1)^* \}$

Sol: $L = \{ \epsilon, 00, 11, 0000, 1111, 0101, 1010, 000000, 001001, 110110, \ldots \}$

Assume L is a CFL

Take $\underbrace{0}_{u}\underbrace{(1)}_{v}\underbrace{1}_{w}\underbrace{0}_{x}\underbrace{1}_{y}$

$uv^2wx^2y = 0(1)^20(1)1 = 01111011 \notin L$

for $i \geq r$, $uv^iwx^iw \notin L$,

Our assumption is wrong

So, $L = \{ ww \mid w \in (0+1)^* \}$ is not a CFL