Theory of Computation Notes

Contributor: Abhishek Sharma

[Founder at TutorialsDuniya.com]

Computer Science Notes

Alphabets :- no empty finite set of symbols is called as Alphabet and it is denoted by 'Σ'

ex. $\Sigma = \{a, b, c, d \ldots z\}$

$\Sigma = \{अ, आ \ldots ह\}$

$\Sigma = \{0, 1\}$ Binary

$\Sigma = \{1\}$ Unary

String :- The sequence of symbol from the alphabet is called as string

ex. $\Sigma = \{0, 1\}$

$w = 0$

$w = 01$

$w = 010$

$w = 102$ // it is not string

Length of The string :- if w is any string define over the alphabet Σ then no. of symbol envolve in the string w is called as length of the string and it is denoted by |w|

ex. $w = \{1, 0\} \in |w| = 2$

$w = \{0, 1, 0\} \in |w| = 3$

$w = \{0, 1, 0, 0\} \in |w| = 4$

Empty string :- A string of length 0

OR

A string without symbol is called empty string and it is denoted by 'ε'

↳ epsailn

ex. $w = |w| = |\epsilon| = 0$

$w = w.\epsilon = \epsilon.w$

$TOC = TOC.\epsilon = \epsilon.TOC$

Sub String :- let u, w be two string define over the alphabet 'Σ' and it is said to be sub string if it is obtained from w as it is the same order.

ex. TOC

T, O, C

TO, OC

TC is not substring because it is not order

Notes

* every string 'ϵ' is a substring of every string
* every string is substring of its ~~string~~ self.
* if u is the substring of w
  then $|u| \leq |w|$

ex. TOC

substrings = TOC, T, O, C, TO, OC, ϵ

Prefix :- Sequence of ~~starting~~ Symbol is called Prefix.

Suffix :- Sequence of ending Symbol is called Suffix.

N = TOC

Prefix = ∈, T, TO, TOC

Suffix = TOC, OC, C, ∈

Language :- Collection of string from the alphabet 'Σ' is called as Language

ex. $\Sigma = \{0, 1\}$.

Language $L = \{00, 01, 10, 11\}$

$L = \{0^n 1^n \mid n \geq 1\} = \{01, 0011, \ldots\}$

Power of N Alphabet :- Let Σ is any alphabet till Σ^a is the set of all the string of length a that is,

$$\Sigma^K = \{w \mid |w| = K\}$$

ex. $\Sigma = \{a, b\}$

$\Sigma^1 = \{a, b\}$

$\Sigma^2 = \Sigma \cdot \Sigma = \{a, b\} \cdot \{a, b\}$

$\{aa, ab, ba, bb\}$

$\Sigma^3 = \Sigma \cdot \Sigma \cdot \Sigma = \{a, b\} \cdot \{a, b\} \cdot \{a, b\}$

$\{aa, ab, ba, bb\} \cdot \{a, b\}$

$\{aaa, aba, baa, bba, aab, abb, bab, bbb\}$

$\Sigma^4 = \{w \mid |w| = 4\}$

$\Sigma^k = \{w \mid |w| = k\}$

Positive Closer :-

'Σ^+'

ex. $\Sigma = \{0, 1\}$

$\Sigma^+ = \{0, 1, 00, 11, 10, 010, 100, 101 \ldots\}$

Clean Closer :-

'Σ^*'

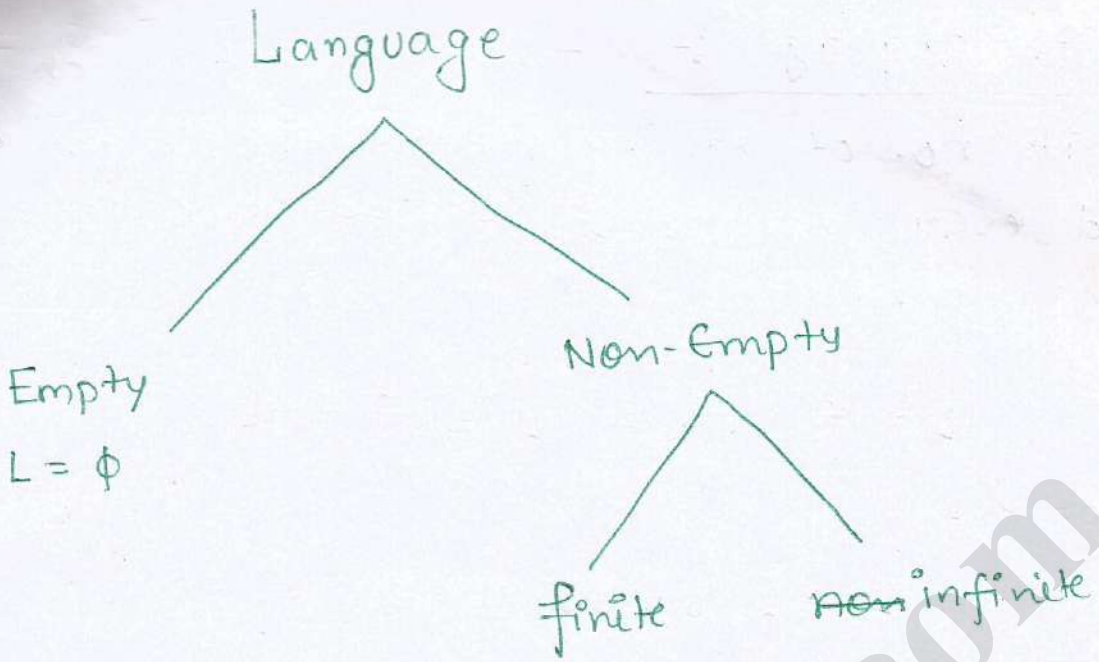
ex. $\Sigma = \{0, 1\}$

$\Sigma^* = \{\epsilon, 0, 1, 00, 11, 10, 010, 100, 101 \ldots\}$

Notes :-

* if Σ is any alphabet then Σ^* is universal language
* if L is any language from the alphabet Σ

$L \leq \Sigma^*$



Empty language :- The Language 'L' that can't Contain any string

$L = \phi \in |L| = 0$

$|L|$ = Number of strings in L

Non empty language :- The language 'L' that Contain at least one string

$L \neq \phi \Leftrightarrow |L| \neq 0$ $\quad L = \{\epsilon\}$

ex. $L = \{a\} \in |L| = 1$

$L = \{\epsilon\} \in |L| = 1$

$L = \{\epsilon, a\} \in |L| = 2$

$L = \{a^n \mid n \geq 0\} \in |L| = \infty$ ~~$\{\epsilon, a, aa, aaa, aaaa, \ldots\}$~~

$L = \{\epsilon\}$ is non empty language

Note :- $\{\epsilon\} \neq \phi$

Finite language :- The language which contains finite no. of string where the length of each and every string is finite is called finite language.

i.e. L = finite = $|L|$ = finite

for ex.

$$L = \{10, 00, 11\}$$

$$|L| = 3$$

$$L = \{101, 110, 010, 1111, 1110\}$$

$$|L| = 5$$

Points :-

$$L = \phi$$

$$|L| = \phi$$

⇒ confusion!

⇒ point to be noted and compare with empty string.

An empty set ϕ is a finite Language

Infinite language :- The language which contains ~~different~~ infinite no. of strings where the length of string is infinite is called infinite language.

for ex.

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

L is infinite

$$|L| = \infty$$

i.e.

$L = \{\Sigma^* + \Sigma^*\}\ \Sigma^+$

$L = \Sigma^*$

both are infinite language

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

$|L| = \infty$ it is infinite language

ex. $L = w \in \Sigma^* \mid |w| =$ even whereas

$\Sigma = \{0,1\}$

$(0,1)^* = \{\epsilon, 0, 1, 00, 01, 10, 11 \ldots$

As per given condn in the example even no. of string

$L = \{\epsilon, 00, 10, 11, \ldots\}$

$L = \{\epsilon, 00, 10, 11, 10, 0000, 1100 \ldots\}$

$|L| = \infty$

Notes :-

Every finite Language is accepted by finite Automata

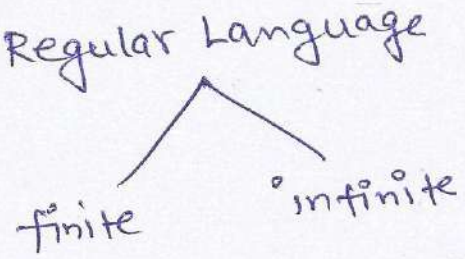
So every finite language is regular language

Every Language

RL ⊂ CFL ⊂ CSL ⊂ REL

Every infinite language can be finite or infinite

RL = finite & infinite

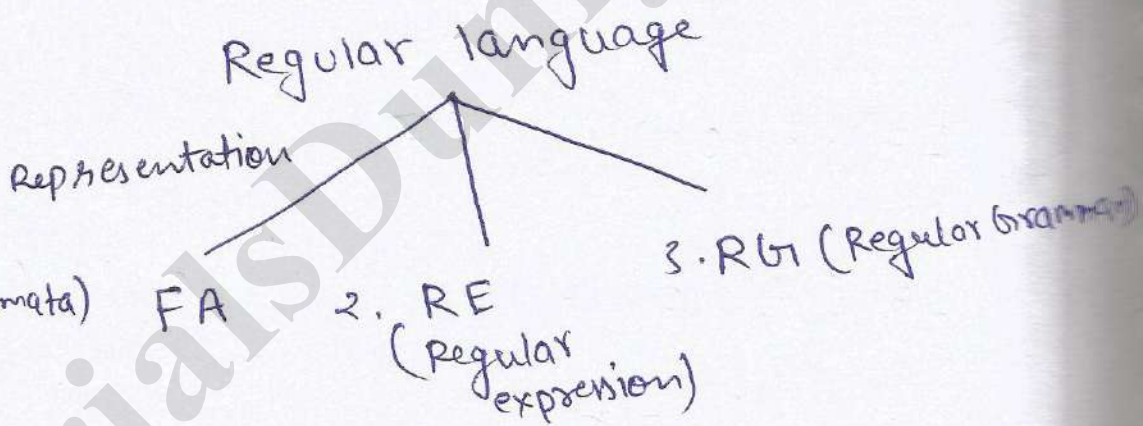


Non Regular Language

↓ always

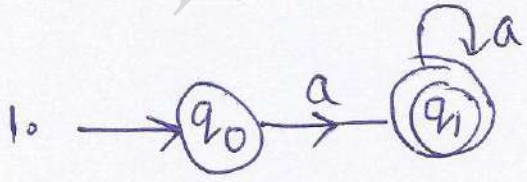
infinite Language

Regular Language :-



RL has 3 Representation FA, RE, RG

ex. Let $L = \{a^n \mid n \geq 1\}$

1. 

= FA

$\delta(q_0, a) \rightarrow q_1$

2. $\gamma = a^+$ = RE

3. $S \rightarrow aS \mid a$ = RG

1. $S \rightarrow a$
2. $S \rightarrow aS$
   $\rightarrow aa$
3. $S \rightarrow aS$
   $\rightarrow aaS$
   $\rightarrow aaa$

Finite Automata :- Mathematical Representation of Regular language is called as finite Automata

OR

Mathematical system that process the string is called as finite Automata

OR

Main Defn :-

Finite Automata is a collection of 5 tuple, (M) → Machine

$= (Q, \Sigma, \delta, q_0, F)$

where Q = Set of all states

Σ = Input Alphabets

δ = $Q \times \Sigma \rightarrow Q$ (Transation function)

q_0 = initial state

F = final state

ID (Instantoneous Description) :- Description movement of FA.

The moments of FA depends on two entities

(A)= current state

(B)= current input Alphabet / symbols

ex. $q_i \xrightarrow{x} q_j$

$\delta(q_i, x) \rightarrow q_j$

(Transfer/ transition/ Configuration)

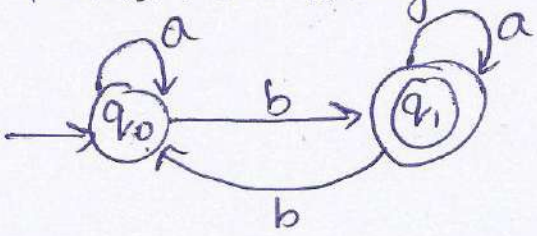
Representation of FA :-

FA can be represented in two ways

(a) Transition Diagram

(b) Transition Table

(a) Transition diagram



IDs :-

$\delta(q_0, a) \rightarrow q_0$

$\delta(q_0, b) \rightarrow q_1$

$\delta(q_1, a) \rightarrow q_1$

$\delta(q_1, b) \rightarrow q_0$

in this eg

$|\Sigma| = 2$ (a, b)

transition state = 2

q_0, q_1

(b) Transition Table

δ / Σ	a	b
$\rightarrow q_0$	q_0	q_1
(q_1)	q_1	q_0

Points :-

* In general finite Automata is ∈FA
* Finit Automata only one initial state
* Finite Automata can be constructed with or without final state ie. no. of final state in FA can be 0 or 1 or more.
* The transfer path for any state in DFA is Unique
* No. of transition state = $|\Sigma|$
* No. of transition in DFA = $|\Sigma| * |Q|$

Q. Construct the language that contains all the string of 0's and 1's where length of string is exactly 2 ?

Sol^n :- $\Sigma = \{0, 1\}$

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 001, 101, 100, 111 \ldots\}$

$\downarrow |w| = 2$

$L = \{00, 01, 10, 11\}$

Q. Construct the language contains all the string of 0's and 1's where the length of string is atmost 2 ?

$\Sigma = \{0, 1\}$

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11 \ldots\}$

$|w| \leq 2$

$L = \{\varepsilon, 0, 1, 00, 01, 10, 11\}$

Q. Construct the Language that contains all the strings of 0's and 1's where every string

a) start with 0

b) start with 10

soln:- $\Sigma = \{0, 1\}$

$\Sigma^* = \{\varepsilon, 0, 1, 00, 10, 11, 01, 101, 000, 010, \ldots$

for condition $\downarrow |w| = 0x$

a) $L = \{0, 00, 01, 000, 001, 011, 010, 0001 \ldots\}$

b) $L = \{10, 101, 100, 1001, 1010, 10001 \ldots\}$

Q. Construct the language that contains all the strings of 0's and 1's where every string

a) start and end with 0

b) start and end with same symbol

c) start and end with different symbol

soln $\Sigma = \{0, 1\}$

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 001, 010, 100, 111, 101 \ldots\}$

$\downarrow |w| = 0x0$

a) $L = \{00, 010, ~~0110, 01110~~ 000, 0110, 0010, 0000, \ldots\}$

$\downarrow |w| = 0 \times 0, 0$

b) $L = \{\overset{0,1,}{0}00, 11, 010, 101, \overset{1\times1,1}{0110}, 1001, 1011, 1101, \ldots$
$000, 111$

$\downarrow |w| = 0 \times 1$

c) $L = \{01, 10, 011, 100, \overset{1\times0}{110}, 001, 1000, 1100, 1110, 1010$

$- - -$

Q. Construct the language that contains all the strings of 0's and 1's where every string

a) where the length of string is even $0 \equiv 0 \pmod 2$

b) odd $1 \equiv 1 \pmod 2$

c) $1 \pmod 3$

Sol b) a) $\Sigma = \{0, 1\}$

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, \overset{000,}{001}, 010, 100, 111, 101, \overset{011}{\ldots}\}$

$|w| =$ ~~$0 \pmod 2$~~ even

$0 \pmod 2$

~~$L = \{\varepsilon, 0, 1, 00, 01, 10, 11, 0000, 0001, 0010, 0100, 1000, 0110, 0111, 1010$~~

$(0, 2, 4, 6, 8 - - -)$

$L = \{w \in \Sigma^* \mid w = 0, 2, 4, 6, 8 - - -\}$

$L = \{\varepsilon, 00, 01, 10, 11, - - - -\}$

$\Sigma = \{0, 1\}$ $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 111 - - - -\}$

b) $|w| =$ odd

$1 \pmod 2$

$(1, 3, 5, 7, 9 - - -)$

$L = \{w \in \Sigma^* \mid w = 1, 3, 5, 7, 9 - - -\}$

$L = \{\varepsilon, 00, 010, 100, 101, 110, 111, 011 - - -\}$

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

(c) $|w| = 1 \pmod 3$

$(1, 4, 7 - - - -)$

$L = \{0, 1, 0000, 0001, 0010, 0100, 1000, 1001, 1010, 1100, 1101, 1110, 1111 - - - -\}$

Q Construct the language that contains all the string of 0's and 1's where

a) The no. of ~~string~~ 0's in string is even

b) The no. of 1's in string is odd

c) No. of 1's in string is multiple of 3

Soln $\Sigma = \{0, 1\}$

$\Sigma^* = \{\epsilon, 0, 0, 1, 00, 01, 10, 11, 100, 010, 101, 001 - - - -\}$

$|w|_0 =$ even

$0 \pmod 2$

$(0, 2, 4, 6, 8 - -)$

$L = \{\epsilon, 00, 11, 1001, 010, 001, - - - -$

b) $\Sigma = \{0, 1\}$

$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 100, 010, 101, 001 - - -\}$

$|w|_1 =$ odd

$1 \pmod 2$

$(1, 3, 5, 7 - - -)$

$L = \{ \varepsilon, 0, 00, 000, 0000, \ldots 1, 0, 111, 1111 \ldots \}$

c) $\Sigma = \{0, 1\}$

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 100, 010, 101, 001 \ldots$
$0000, 0001, 0010, 0100, 1000, 1001, 1010, 1100, 1101, 1110, 1111$

$|w|_1 =$ 0 Multiple of 3

0(Mod 3)

$L = \{ \varepsilon, 0, 00, 111, 000, \ldots \}$

Q Construct the language that contain's all the string of 0's and 1's where all the string is palladrome

$\Sigma = \{0, 1\}$

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 100, 010, 101 \ldots \}$

$L = \{ w \in \Sigma^* \mid w = w^R \}$

$L = \{\varepsilon, 0, 1, 00, 11, 101, 010, 000, 111 \ldots \}$

Regular Expression :- if Σ is any alphabet than the expression which is constructed over the Symbol of Σ using operator $*, +, .$ is called Regular expression.

A set $*$ is said to be klean clouser, $+$ is positive klause and $.$ is Contakanation

these three operator is said to be regular operator and expression that generate regular language is called regular expression.

ex. Let us $\Sigma = \{0, 1\}$

$R = 0, 1$

$R = 0^* 1$

$R = 0^+ 1$

$R = 1^* 0^*$

$R = (01)^*$

$r = 0^* + 1^* + (01)^*$

Points :- 1. Regular expression always reference regular language.

for ex. $r = 0.(0+1)$

$L = (00+01)$

2. Arthimatic expression always reference numerical values.

ex. $x = 0.(0+1)$

3. Regular expression

if R is any regular expression L(R) is a regular language generated by

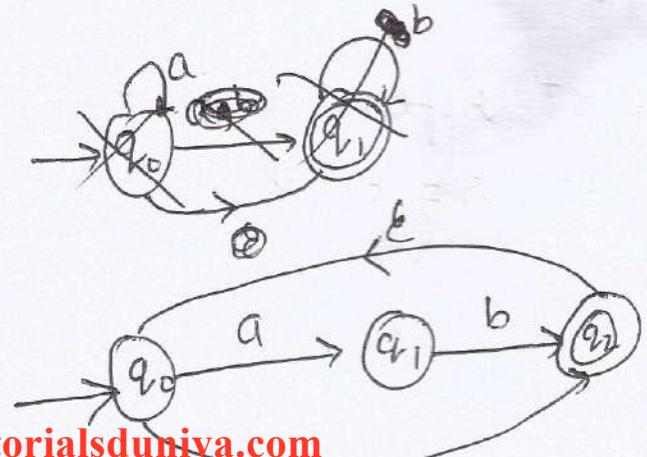
ex.	R (RE)	RL
1.	$r = \phi$	$L = \phi$
2.	$r = \epsilon$	$L = \{\epsilon\}$
3.	$r = a$	$L = \{a\}$
4.	$r = ab$	$L = \{ab\}$
5	$r = a+b$	$L = \{a, b\}$
6.	$r = w_1 + w_2 + \dots w_n$	$L = \{w_1, w_2, \dots w_n\}$
7.	$r = a.(a+b)$	$L = \{aa, ab\}$
8.	$r = a^*$	$L = \{\epsilon, a, aa, aaa - - - -\}$
9.	$r = a^+$	$L = \{a, aa, aaa - - - -\}$

Regular Operator precedence

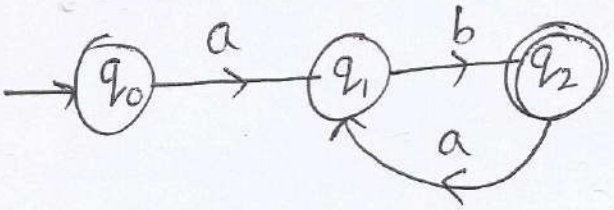
1 *
2 .
3 +

Deductions :-

1. every finite language is regular.
2. $r = (ab)^* \Rightarrow L = \{\epsilon, ab, abab, ababab - - - \}$



3. $r = (ab)^+ \Rightarrow L = \{ab, abab, ababab - - - - - -\}$



if r is regular expression that r^* and r^+ is also regular expression.

$r^* = \{\varepsilon, r, rr, rrr - - -\}$

$r^+ = \{r, rr, rrr - - -\}$

$L(r^*) = [L(r)]^*$

$L(r^+) = [L(r)]^+$

ex. $r = ab \Rightarrow L = \{ab\}$

$r^* = (ab)^* \Rightarrow L = \{\varepsilon, ab, abab - - -\}$

$r^+ = (ab)^+ \Rightarrow L = \{ab, abab, ababab - - -\}$

4. if $r = \varepsilon$ than $r^* = \{\varepsilon\}$

$r^+ = \{\varepsilon\}$

if $r = \phi$ than $r^* = \varepsilon$

$r^+ = \phi$

In general both r^*, r^+ reference is finite language except for $r = \phi$ or $r = \varepsilon \Rightarrow r^*, r^+$ represents of finite language

5. $r^* = r^+ \cup \varepsilon \Rightarrow r^+$ is subset of r^* ✓

i.e. $r^+ + r^* = r^*$

$r^+ \cap r^* = r^+$

6. (i) $r^* . r = r^+ = r . r^*$

(ii) $r^+ . r^* = r^+ = r^* . r^+$

(iii) $r^* . r^* = r^*$

(iv) $r^+ + \varepsilon = r^*$

(v) $(r^*)^* = r^*$

(vi) $(r^+)^* = r^*$

(vii) $(((r^+)^*)^+)^{*/+} = r^*$

7. if r_1, r_2 with the regular expression both $(r_1 + r_2)$, $(r_1 . r_2)$ is also regular expression

i.e. $L(R_1 + R_2) = L(R_1) + L(R_2)$

$L(R_1 . R_2) = L(R_1) . L(R_2)$

8. if R_1, R_2 two regular expression then

$(r_1 + r_2)^* = (r_1^* + r_2^*)^*$

$= (r_1^* + r_2)^*$

$= (r_1 + r_2^*)^*$

$= (r_1^* . r_2^*)^*$

9 if r_1, r_2 be two regular expression then

$$(r_1 \cdot r_2)^* \cdot r_1 = r_1 \cdot (r_2 \cdot r_1)^* \quad \text{// shifting Rule}$$

10. $r_1 = (a^* + b^*)^* = (a^* \cdot b^*)^*$

$r_2 = (a^* + b^+ a^*)^*$

Justify $r_1 = r_2$

$r_1 = (a^* + b^*)^*$

$= (a^* \cdot b^*)^*$

$r_2 = (a^* + b^+ a^*)^*$

$(a^* + b^+)^*$

$(a^*)^*$

a^*

soln :- $r_2 = (a^* + b^+ a^*)^*$

$(a^* \epsilon + b^+ a^*)^*$

$(a^* [\epsilon + b^+])^*$

$(a^* b^*)^*$

$r_1 = r_2$

two regular expression are equal if and only if

$$L(r_1) = L(r_2)$$

Points

ex. $r = 0^*$

$L = \{\epsilon, 0, 00, 000 \ldots\}$

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

ex. $r = 0^* . 1$

$L = \{0^n . 1 \mid n \geq 0\}$

ex. $r = (01)^*$

$L = \{\epsilon, 01, 0101, 010101 \ldots\}$

$L = \{01^n \mid n \geq 0\}$

Q. $r = 0^* 1^*$

$L = \{\epsilon, 0, 00, 000 \ldots, 1, 11, 111, \ldots\}$

$L = \{0^{m,n} \mid m,n \geq 0\}$

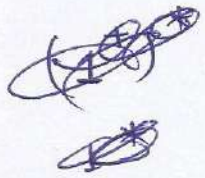
Q. $r = 0^+ 1^+$

$L = \{0^m 1^n \mid m,n \geq 1\}$

Q $r = 0^* 1^+$

$$L = \{ 0^m, 1^n \mid \begin{matrix} m \geq 0 \\ n \geq 1 \end{matrix} \}$$

Q $r = (0^* 1^+)^*$



$L = \{ 0^m 1^n \mid \}$

$L = \{ (0^m, 1^n)^p \mid$ $m \geq 0, n \geq 1$ $p \geq 0 \}$

Q Construct RE that contain all string 0's and 1's

a) including ϵ

b) excluding ϵ

$\Sigma = \{0, 1\}$

$r = (0+1)^*$

b) $r = (0+1)^+$

Q Construct RE that contain all string 0's and 1's

a) where every string start with 0

b) Start with 10

c) Start with 011

L = {0, 01, 00, 011, 010, 000, - - - - }

a) $r = 0.(0+1)^*$

b) $r = 10.(0+1)^*$

c) $r = 011.(0+1)^*$

Q String end with 10

end with 011

end with 1010

a) $(0+1)^*.10$

b) $(0+1)^*.011$

c) $(0+1)^*.1010$

Q where every string contain substring

a) 101 → $(0+1)^*\ 101\ (0+1)^*$

b) 0101 → $(0+1)^*\ 0101\ (0+1)^*$

Q where the no. of 0's in string is

a) excatly 2

b) at most 2

c) even 0,2,4,6,8

d) odd (1,3,5,7,9)

e) At least 2

Soln :- $\Sigma = \{0,1\}$ $|w|_0 = 2$

x 0 x 0 x

↓

a) $r = 1^* 0.1^* 0.1^*$

$|w|_0 = 0,1,2$

b) $r = (1^* + 1^* 0.1^* + 1^* . 0.1^* . 0.1^*)$

$|w|_0 = 2,4,6,8$

c) $r = (1^* . 0.1^* . 0.1^*)^* + 1^*$

(d) $|w|_0 = 1,3,5,7$

$(1^* 0.1^*) . (1^* 0.1^* . 0.1^*)^*$

(e) $|w|_0 = 2,3,4,5, \ldots$

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

Q Construct RE for the following Language

$L = \{0^m 1^n \mid m+n = \text{even}\}$

$L = \{0^m 1^n \mid m+n = \text{odd}\}$

Soln $\Sigma = \{0, 1\}$

even

$m = 2x$, $n = 2x$ — m+n = even

$m = 2x+1$, $n = 2x+1$ — m+n = even

$0^{2x} 1^{2x}$ — $0^{2x+1} 1^{2x+1}$

$(00)^x (11)^x$ — $(00)^x \cdot 0\ (11)^x \cdot 1$

$(00)^* (11)^*$ — $(00)^* \cdot 0\ (11)^* \cdot 1$

$r = (00)^* (11)^* + (00)^* \cdot 0 (11)^* \cdot 1$ ✓

(b) $\Sigma = \{0, 1\}$

odd

$m = \text{odd} = 2x+1$, $n = \text{even} = 2x$

$m = \text{even} = 2m$, $n = \text{odd} = 2m+1$

$(00)^x \cdot 0\ (11)^x + (00)^x (11)^x \cdot 1$

$(00)^* \cdot 0\ (11)^* + (00)^* (11)^* \cdot 1$

Soln $0^{2x}\,1^{2x+1}$ $\qquad$ $0^{2x+1}\,1^{2x}$

$(00)^x \cdot (11)^x \cdot 1$ $\qquad$ $(00)^x \cdot 0 \cdot (11)^x$

$(00)^* (11)^* \cdot 1 + (00)^* \cdot 0 \cdot (11)^*$ A

Q Construct RE for the language

$L = \{ w x w^R \mid w, x \in (0+1)^+ \}$

Soln :- $w = 100 \Rightarrow 100 \times 001 \Rightarrow 1 \times 1$

$w = 001 \Rightarrow 001 \times 100 \Rightarrow 0 \times 0$

$w x w^R = 0 \times 0 + 1 \times 1$

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

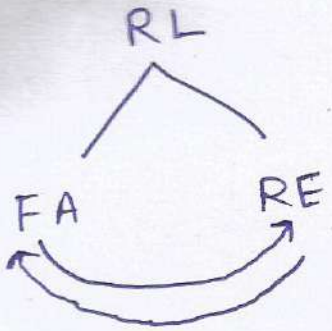
TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Equivalance b/w FA and RE :-



There are two techniques for the conversion of FA TO RE and RE TO FA

1. ARDEN'S Theoram
2. State ELIMANATION Method

1. ARDEN'S LEMMA Theoram :-

This Mechanism can be used for DFA and NFA and Can't be used for ε NFA.

if P, Q, R be the three RE i.e.

$r = Q + RP$ than

(a) $r = Q + RP$ has unique solution if P does not contains ε that is $r = Q + RP \Rightarrow \boxed{r = QP^*}$

(b) if P contains ε than $r = Q + RP$ has infinit many solution

Ex:

$$A = Aa + \varepsilon \quad \text{---------- (i)}$$

$$A = \varepsilon + Aa$$

$$R = Q + RP$$

$$R = QP^*$$

$$A = \varepsilon a^*$$

$$A = a^*$$

$$B = \cancel{B\alpha + \beta a} \quad Ab + Ba$$

$$A\beta = \alpha^* b + B a$$

$$\beta = Q + RP$$

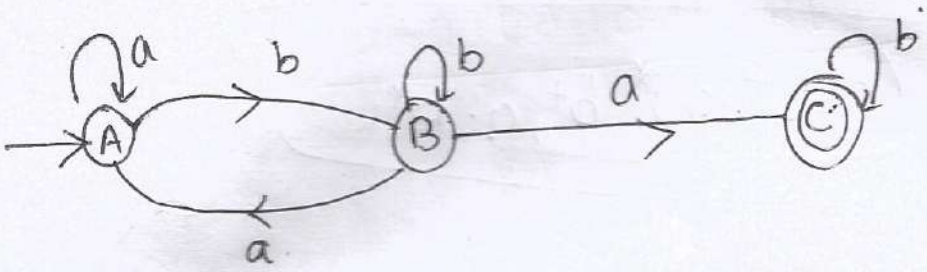
$$B = a^* b a^*$$

$$C = Bb + Ca + cb$$

$$C = a^* b a^* b + C(a+b)$$

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

ex.



$A = Aa + Ba + \epsilon$ ——— (1)

$B = Ab + Bb$ ——— (2)

~~$B = a^*b + Bb$~~

~~$B = a^*b.b^*$~~

$C = (A+B)$

~~$C = Ba + Ca + Cb$~~

~~$= a^*b.b^*.b + \epsilon(a+b)$~~

$a^*b.b^*.a + Cb$

~~$C = a^*b.b^*.a(b)^*$~~

~~$C = a^*b.b^*.a\,b^*$~~

$C = Ba + Cb$

$R = Q + \phi$

$$C = (a + b^+ \cdot a)^* \, b^+ \cdot a \cdot b^*$$

Soln :-

$$A = Aa + Ba + \epsilon \quad \text{——— (1)}$$

$$B = Bb + Ab$$

$$B = Ab + Bb$$

$R \quad Q \quad RP$

$\Rightarrow B = QP^*$

$= Ab \cdot b^*$

$= Ab^+$

$b(\epsilon, b, bb$

$(b, bb, bbb$ —

b^+

$B = (a + b^+ a)^* \cdot B$

$C = Ba + Cb$ — (3)

$C = \underbrace{(a + b^+ \cdot a)^*}_{Q} \cdot \underbrace{b^+ a}_{P} + Cb$

$C = (a + b^+ \cdot a)^* \cdot b^+ \cdot b$

b^*

$A = Aa + (Ab^+)a + \epsilon$

$A = A(a + b^+ a) + \epsilon$

$R \quad R \quad P \quad Q$

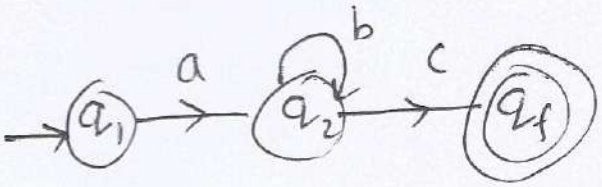
$A = QP^*$

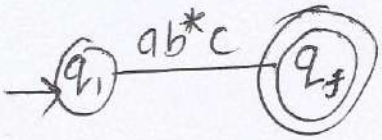
$= \epsilon \cdot (a + b^+ \cdot a)^*$

$A = (a + b^+ \cdot a)^*$

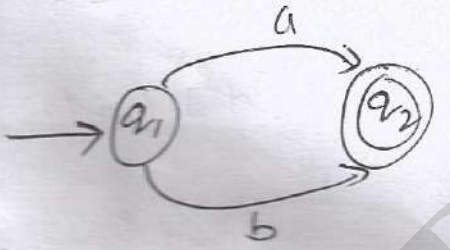
2. STATE ELIMINATION METHOD :-

Ex.

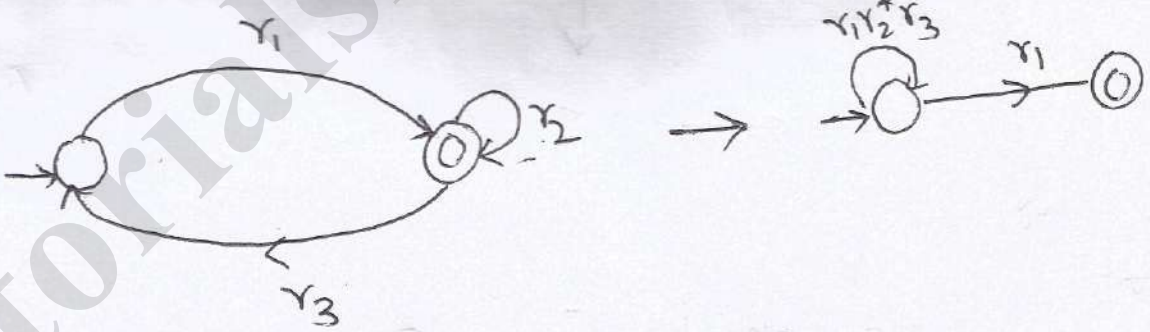


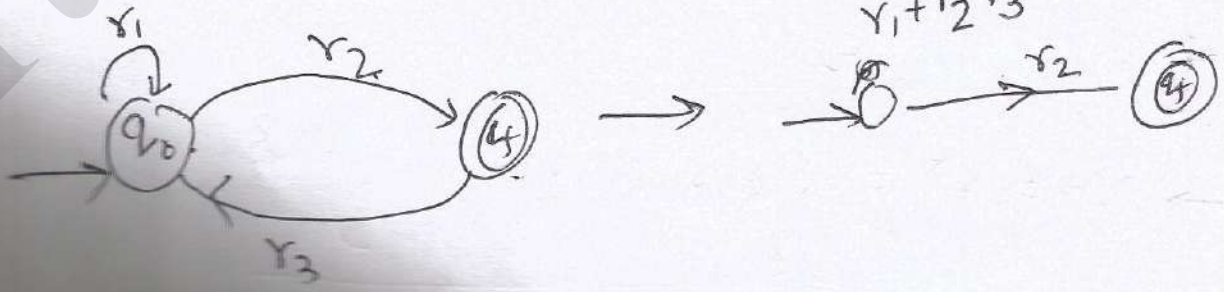


a+b is represented by



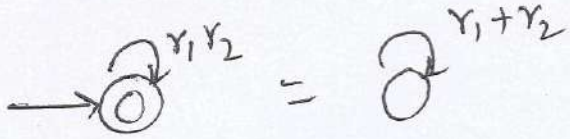
ex.





untill following ~~who~~ you reach this is final

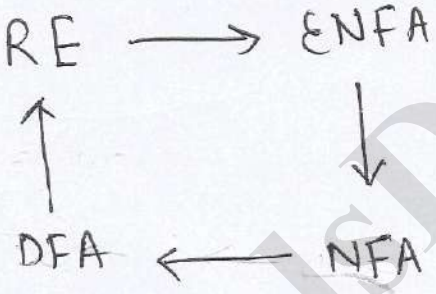
(1)  $\Rightarrow r = r^*$

(2) 

(3) 

(4) 

FA ⟷ RE

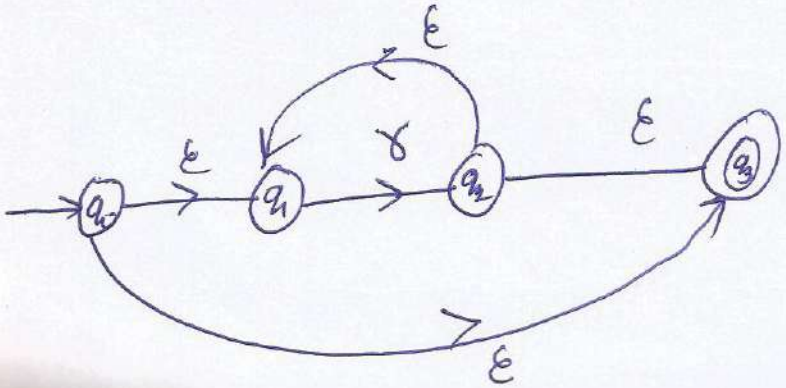


~~Equi~~

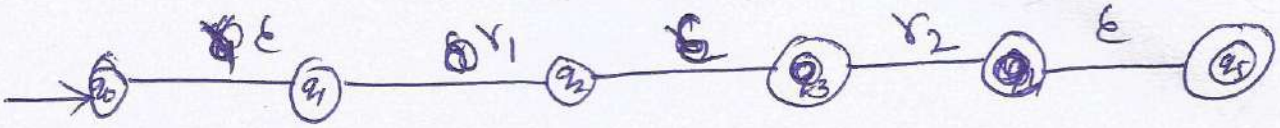
Method of Synthesis :-

i) Klean closure

Rule → $re = r^* = \{\varepsilon, r, rr, rrr \text{ - - - }\}$

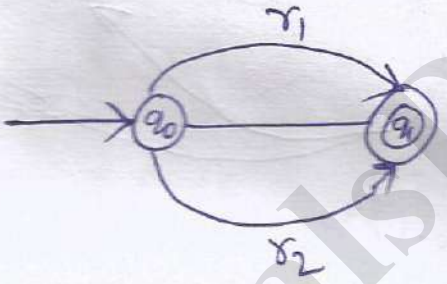


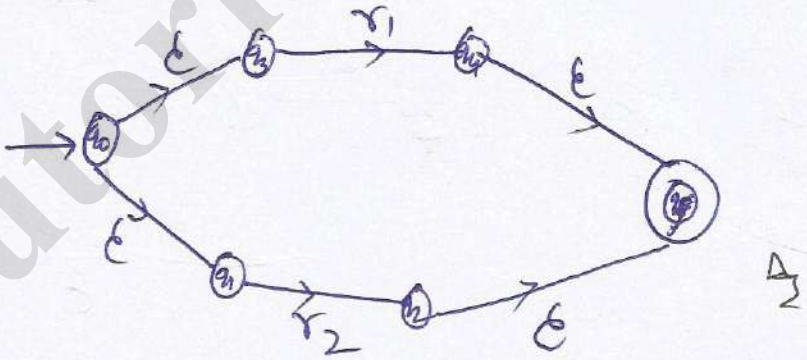
(ii) Concatenation $(r_1 \cdot r_2)$



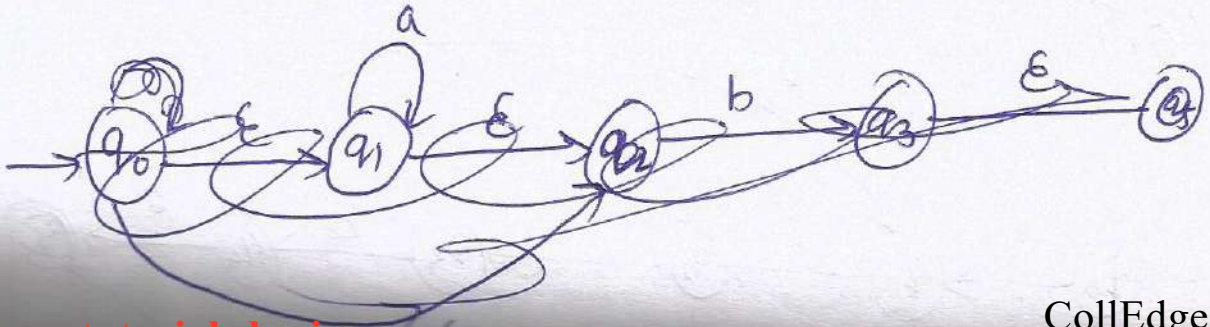
(iii) Union (U)

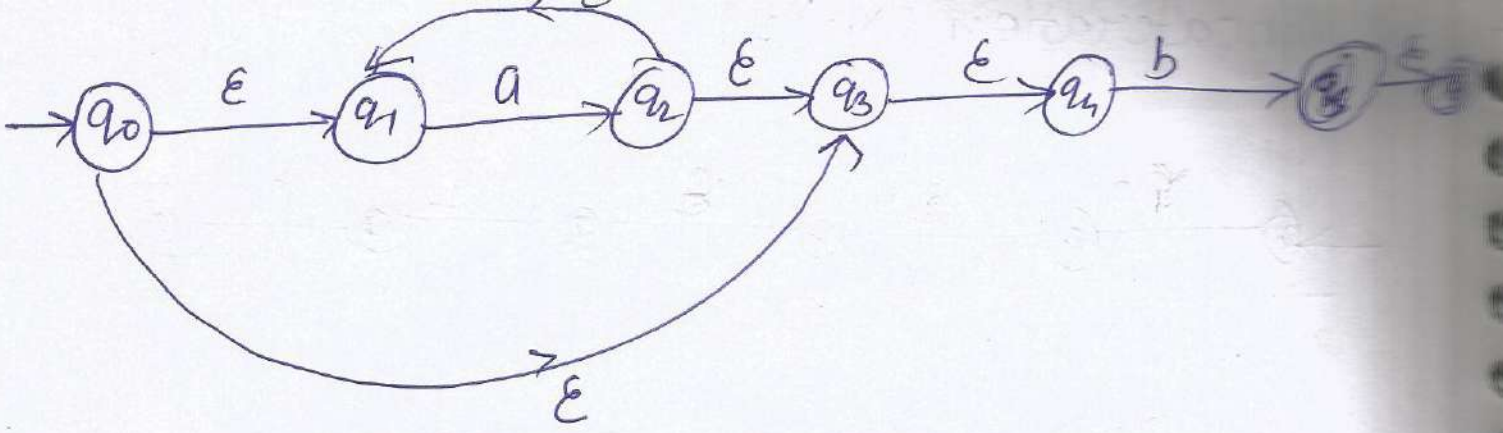
$r_1 + r_2 = (r_1 \cup r_2)$



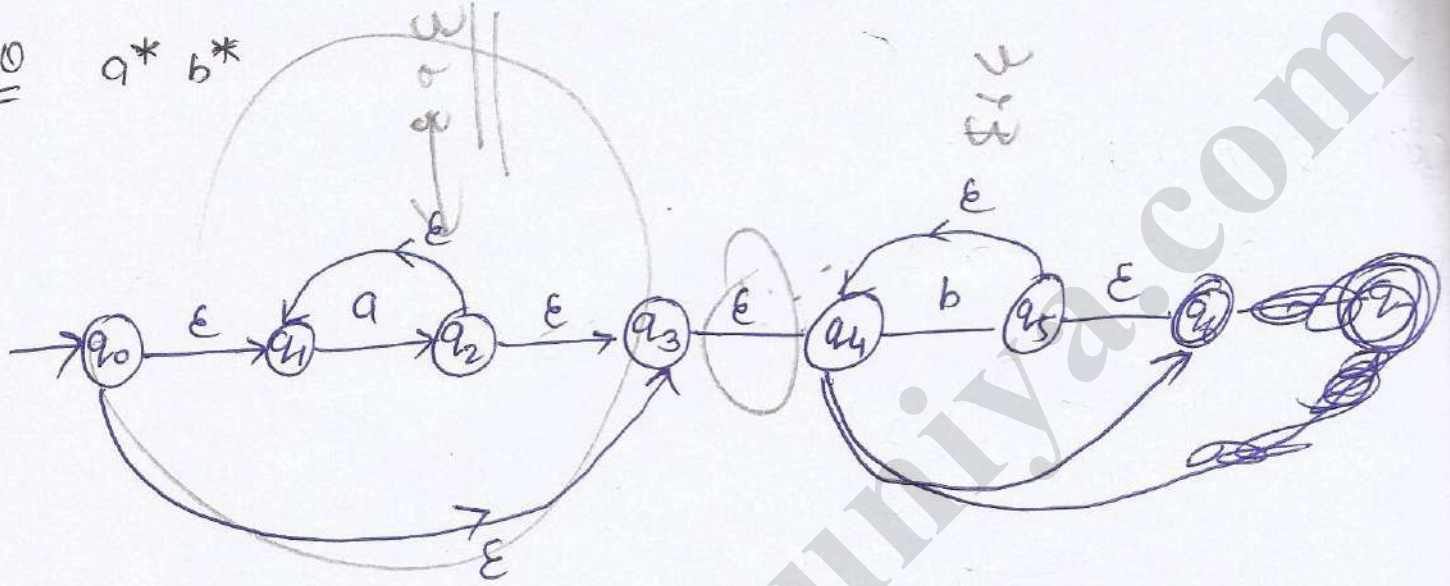


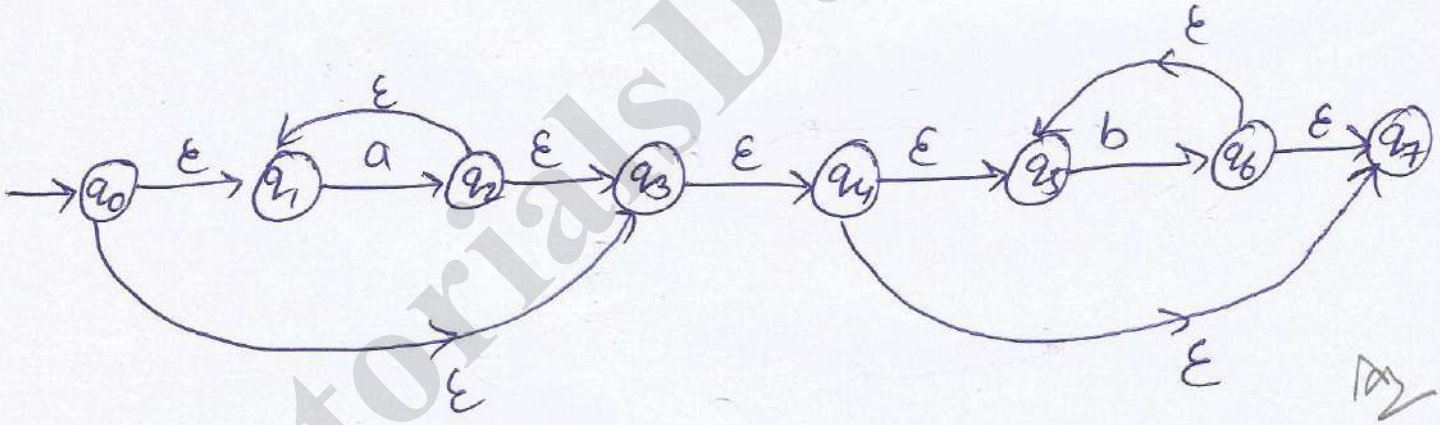
Ex. a* b



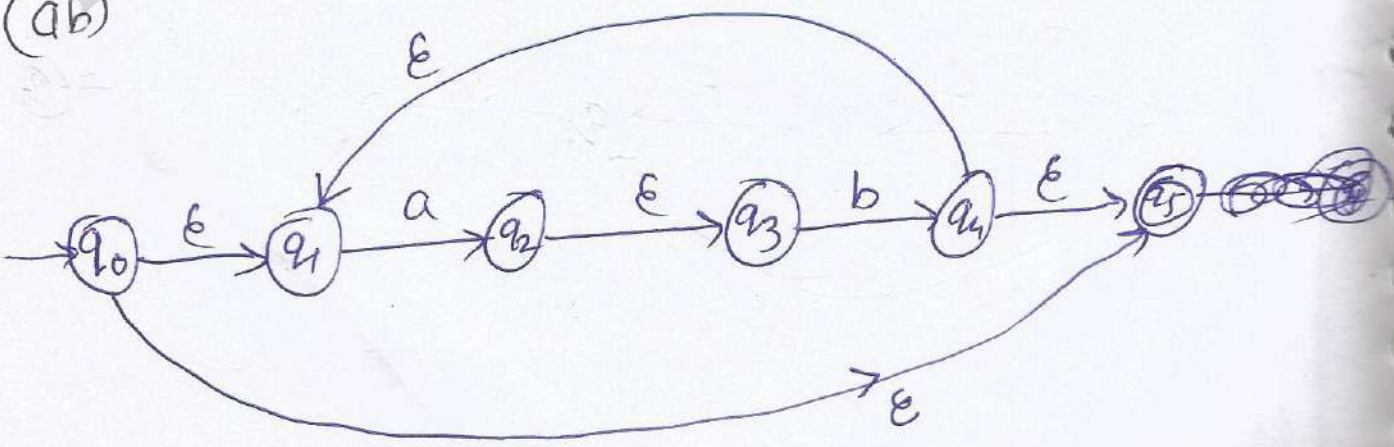


Q. $a^* b^*$

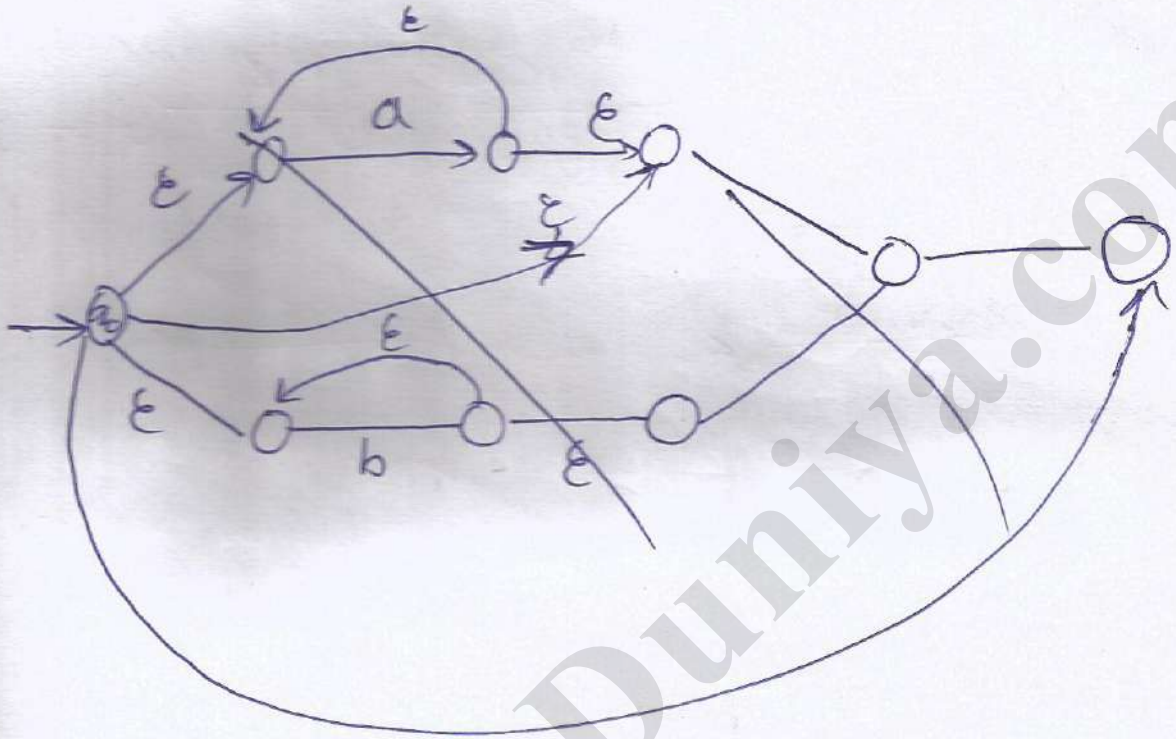


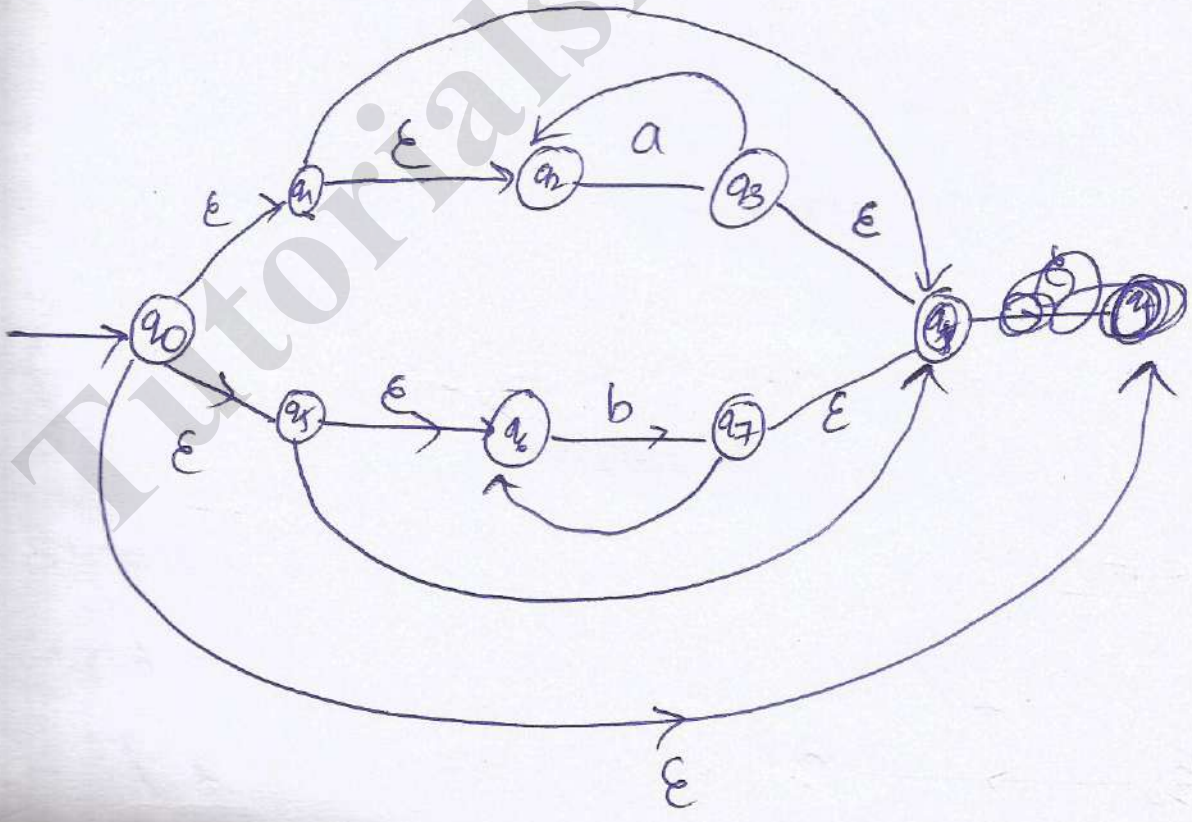


Q. $(ab)^*$



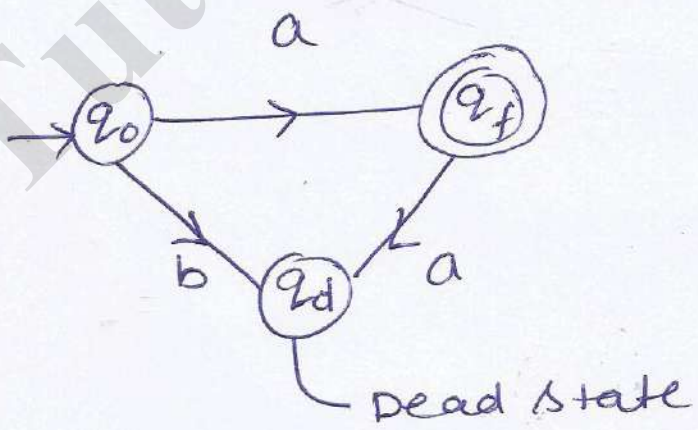
Q $a^* + b^*$





$\equiv (a^* + b^*)^*$

Dead State :-



where NO outgoing edges

Closure properties of Regular Language :-

Regular language satisfy the clouse property under the following operator.

i) Complement
ii) Kleen Closure
iii) Positive closure
(iv) Reversal operator
(v) prefix operator
(vi) Union operator($L_1 \cup L_2$)
(vii) Intersection operator($L_1 \cap L_2$)
(viii) Concatenation ($L_1 \cdot L_2$)
(ix) Difference operator($L_1 - L_2$)
(x) Symmetric difference ($\frac{L_1}{L_2}$)
(xi) Quotient operator
(xii) Substitution

(iv) Reverse/Reversal operator :- if L is a Regular Language then L^R is also a Regular language

Process to get finite Automata for L^R :-

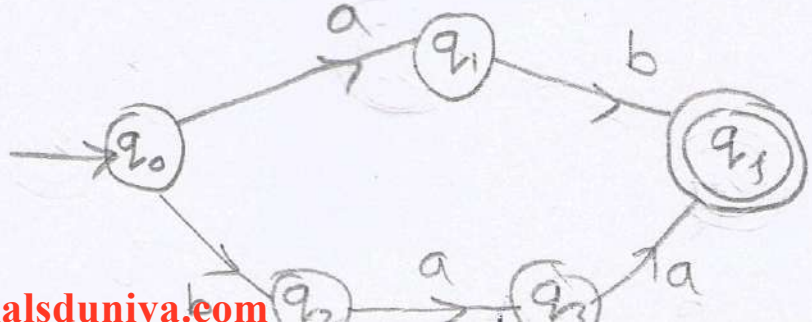
Step 1 :- Interchange initial and final states

Step 2 :- Change the direction of arrow (Adjust)

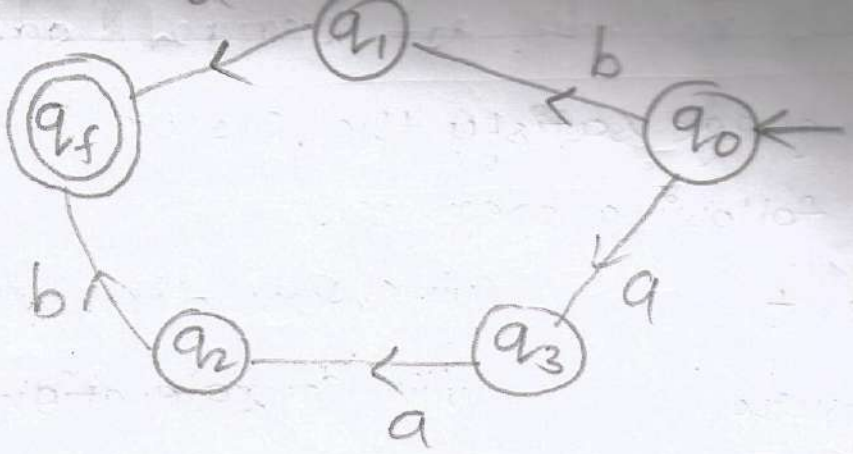
Step 3 :- If more than one initial state occur then convert the system to have only one initial state.

ex. $L = \{ab, baa\}$

F.A. =



L^R =



Prefix operator :- if L is a Regular Language than prefix of L^R is also a Regular language

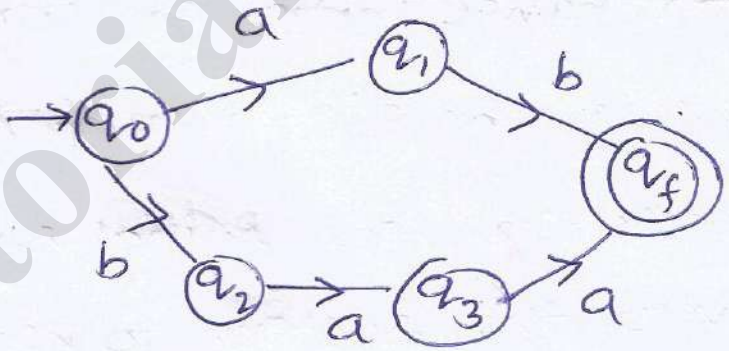
Process to get FA for prefix of L :-

Step 1 :- In Case of NFA make all the state as final state

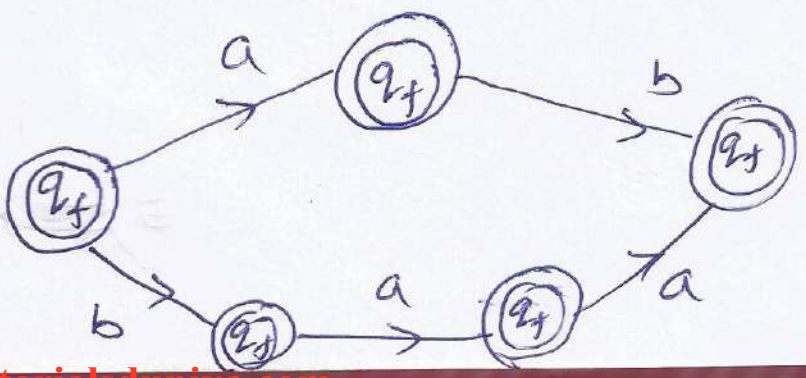
Step 2 :- In Case of DFA EXcept dead state and make the all state as final

ex. L = {ab, baa}

F.A. =



Prefix of L =



xi) Quotient operator :- if L_1 and L_2 be Regular Language than $\frac{L_1}{L_2}$ is also Regular language

where

$$\frac{L_1}{L_2} = \left\{ x \mid xy \in L_1, y \in L_2 \right\} \Rightarrow \frac{xy}{y} = x$$

$$\frac{L_1}{L_2} = \left\{ y \mid xy \in L_1, x \in L_2 \right\} \Rightarrow \frac{xy}{x} = y$$

ex. $\frac{010}{10}$ Result = 0

$\frac{0101}{10} = 01$

$\frac{101}{00} = \phi$

$\frac{100}{100} = \epsilon$

Pumping lemma for Regular Language

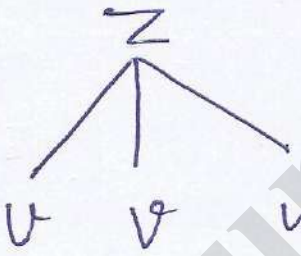
process of pumping lemma :-

Step 1 :- Assume L is regular ⇒ L satisfy pumping Lemma property

Step 2 :- Choose $z \in L$ such that $|z| \geq n$

Step 3 :- Split z into three part such that

$|uv| \leq |z|$ and $|v| \neq 0$



Step 4 :- if there exist at least one value of L such that $uv^iw \notin L$ that means u, v^i, w does not belongs to L.

if L does not satisfy pumping lemma property which is contradiction.
L is non regular Language.

ex. $L = \{a^n b^n \mid n \geq 0\}$ Prove this is non regular language

soln :- Step 1 :- Assume L is regular language

Let $z \in L$

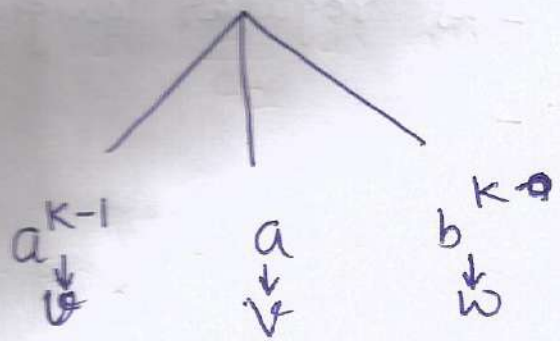
TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

$z = a^K . b^K$



$uv^iw = a^{k-1} a^i b^K$ for $i = 2$

$a^{K+1} . b^K \notin L \quad [\neq a^k b^k]$

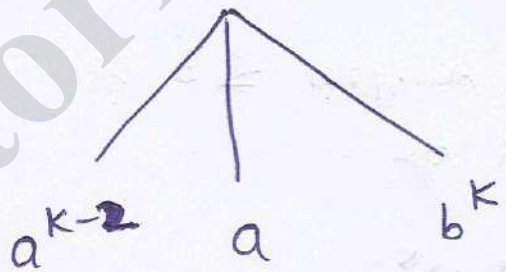
$\Rightarrow L = \{a^n b^n \mid n \geq 0\}$ is not regular language

Q. $L = \{a^m b^n \mid m < n\}$

Step 1:- Assume L is regular language

Let $z \in L$

$z = a^{K-1} b^K$



$uv^iw = a^{k-2} . a^i . b^K$

for $i = 2$

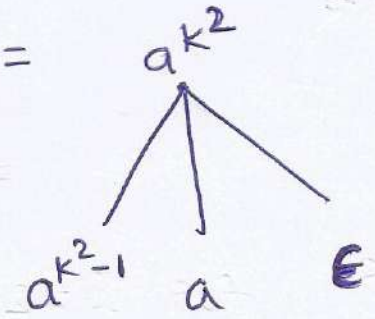
$a^K b^K \notin L \quad [\neq a^{k-1} . b^k]$

$\Rightarrow L = \{a^m b^n \mid m < n\}$ is a not a regular language

Q. $L = \{a^{n^2} \mid n \geq 0\}$

Soln:- Step 1:- Assume L is regular language

Step 2:- Let $z \in L$

$z =$ 

$uv^iw = a^{k^2-1} \cdot a^i \cdot \epsilon$

for $i=2$

$a^{k^2+1} \cdot \epsilon$

$a^{k^2+1} \notin L$

L does not satisfy pumping lemma property which is contradiction.

L is not a regular language.

Q. Number of string length ≤ 3 generated by following RE will be - - - - -

$RE = (1+01)^* \; 0 \; (1+0)^*$

Soln:- $L=0 \Rightarrow 0$

$L=1 \Rightarrow 0$

$L=2 \Rightarrow 00, 01, 10,$

$L=3 \Rightarrow 001, 011, 010, 000, 100, 010, 101$

$L = \{0, 00, 01, 10, 001, 011, 010, 000, 100, 101, 010\}$

Q. if $A = (0^* + 10)^*$

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

a) A C B b) B C A c) A = B d) None of these

$[(R_1 + R_2^*)^* = (R_1^* + R_2)^*]$

Q. i) 0^*

ii) $(00)^*$

iii) $(0+\varepsilon)\cdot 0^*$

iv) $0.(00)^*$

i) 0^* = iii) $(0+\varepsilon)\cdot 0^*$

Q. Find the sets represented by following regular expression

i) $(a+b)^*(aa+bb+ab+ba)^*$

The set of all string having odd no. of symbols from $\Sigma^* = \{a, b\}$

ii) $(a+b)\cdot(a+b)^*$

$\{a, b, ba, bb, bab, bba, baa, bbb - - - -\}$

Eqvivalance b/w DFA and NDFA

Conversion NFA TO DFA

Q. Construct Deterministic Automata eqvivalant to

$M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_0\})$

where δ is define by its state table transition

δ / Σ	0	1
$\rightarrow q_0$	q_0	q_1
(q_1)	q_1	q_1, q_0

Soln 1. for the deterministic Automata M_1, the states are the subsets of $\{q_0, q_1\}$

i.e $\phi, [q_0], [q_0, q_1], [q_1]$

2. q_0 is the initial state

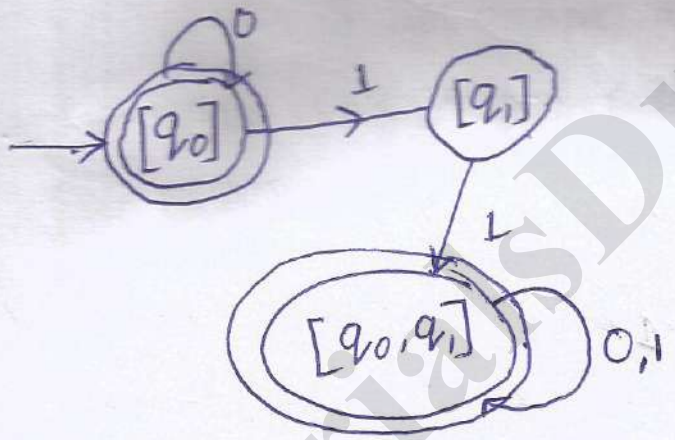
3. q_0 and q_0, q_1 are the final states Reason these are the the only state containing q_0 (which is final state in the given table)

4. δ is difined by the state table given belong

Q / Σ	0	1
ϕ	ϕ	ϕ
$[q_0]$	$[q_0]$	$[q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
$[q_1]$	q_1	$[q_0, q_1]$

$\delta([q_0], 0) \longrightarrow [q_0]$

Construct DFA To given Table



Q. Construct DFA equivalent to

$M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$

where δ is given by the table

Q / Σ	a	b
→ q_0	q_0, q_1	q_2
q_1	q_0	q_1
q_2		q_0, q_1

1. For the deterministic Automata M_1, the states are the subsets of $\{q_0, q_1, q_2\}$

$\phi, [q_0], [q_0, q_1], [q_1]$

2. q_0 is the initial state

δ / Σ	a	b
ϕ	ϕ	ϕ
$[q_0]$	$[q_0, q_1]$	q_2
$[q_0 q_1]$	$[q_0, q_1]$	$[q_1, q_2]$
$[q_1]$	$[q_0]$	$[q_1]$
$[q_2]$		$[q_0, q_1]$
$[q_1, q_2]$	$[q_0,]$	$[q_0, q_2]$
$[q_0, q_1, q_2]$	$[q_0, q_1]$	$[q_0 q_1, q_2]$
$[q_0, q_2]$	$[q_0 q_1]$	$[q_0, q_2, q_2]$

$2^Q = 2^3 = 8$

Grammar

A grammar is collection of 4 variables or tuples

ie. (V_n, Σ, P, S)

→ where V_n is a finite non empty set who's element are called variable

→ Σ is a finite non empty set who's elements are called as Terminals. (Small letters)

$\because V_n \cap \Sigma = \phi$

→ S is a special variable (i.e. an element of V_n called the start symbol)

→ P is a finite set who's elements are $\alpha \to \beta$ where α and β are strings on $V_n \cup \Sigma$

α has at least one symbol from V_n

The elements of P are called productions or production rule or rewritting rule.

Derivations and the language generated by the Grammar :-

Q. if $G = (\{S\}, \{0,1\}, \{S \to 0S1, S \to \wedge\}, S)$ ($\wedge$ = NULL)

find $L(G) = ?$

Sol^n:- $S \to \wedge$ is the production so $\wedge$ is in L of G

also for $\forall\, n \geq 1$

$S \to 0S1$

$S \to 00S11$

$S \to 000S111$

$S \to 0^n 1^n$

$\therefore \quad 0^n 1^n \in L(G)$ s.t. $n \geq 0$

description:- The derivation of w starts with S

if $S \to \wedge$ is applied first, we get $\wedge$

In this case w = $\wedge$.

The first production to be applied $S \to 0S1$

at any stage if we apply $S \to \wedge$ we get a terminal string. A terminal string is obtain is only by applying $S \to \wedge$

Thus the derivation of w is of the form

$S \to 0^n S 1^n \to 0^n 1^n \; \forall\, n \geq 1$

i.e. $L(G) = 0^n 1^n \in L(G)$ s.t. $n \geq 0$

Q Let G = ({S, C}, {a, b}, {S → aCa, C → aCa | b}, {S})

find L(G) = ?

Soln :- S → aCa , C → aCa | b

S → aaCaa by application C → aCa

S → aaaCaaa

⋮ ⋮

S → $a^n C a^n$ by Application C → aCa for (n−1) times

S → $a^n b a^n$ $n \geq 1$

$L(G) = a^n b a^n$ Ans

Description :-

Q $G = (\{S, a_1\}, \{0,1,2\}, \{S \to 0Sa_12, S \to 012, 2a_1 \to a_12, 1a_1 \to 11\}, \{S\})$

Soln:- $S \to 0Sa_12$

$S \to 00\underline{12a_1}2$ by application $S \to 012$

$S \to 001a_12$ by applicati

$S \to \underline{00112}$

$S \to 0\underline{00112}2a_12$

$00011\cancel{2}a_122$

~~000111222~~ $0001\underline{1a_1}222$

000111222

0

$L(G)$ $0^n1^n2^n$ $n \geq 1$ A^*

Q. $G = (\{S, A_1, A_2\}, \{a, b\}, P, S)$

where P consists of

$S \to aA_1A_2a$, $A_1 \to baA_1A_2b$

$A_2 \to A_1ab$, $aA_1 \to baa$

$bA_2b \to abab$

Test w, $w = baabbbabaaabbaba$

Sol^n:-

$S \rightarrow aA_1A_2a$

~~$S \rightarrow abaA_1A_2bA_2a$~~

~~$S \rightarrow abaA_1A_1abbA_1aba$~~

~~$S \rightarrow abaabbaabaA_1A_2babbA_1aba$~~

~~$S \rightarrow a baaA_1aba$~~

~~$babaaaaba$~~

~~$abaA_1A_2bA_2a$~~

$S \rightarrow baaA_2a$ (by using $aA_1 \rightarrow baa$)

$S \rightarrow baaA_1aba$ (by using $A_2 \rightarrow A_1ab$)

$S \rightarrow baabaA_1A_2baba$ (by using $A_1 \rightarrow baA_1A_2b$)

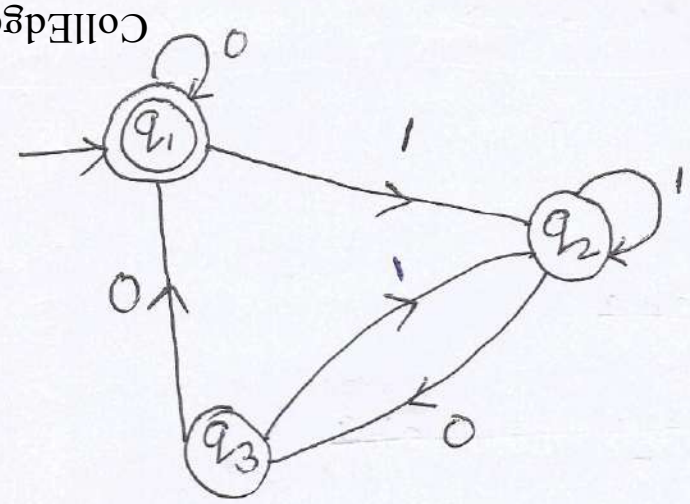
$S \rightarrow baabbaaA_2baba$ (by using $aA_1 \rightarrow baa$)

$S \rightarrow baabbaaA_1abbaba$ (by using $A_2 \rightarrow A_1ab$)

$S \rightarrow baabbabaaabbaba$ (by using $aA_1 \rightarrow baa$)

Accepted

Q.



Construct RE

Σ / Q	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_1	q_2

use ARDENS Theoram :-

$q_1 = q_1 0 + \varepsilon$ ——— (1)

(R, Q, P)

~~$q_1 = q_p 0$~~

$q_1 = \varepsilon 0^*$

$q_1 = 0^*$ ——— (2)

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

$q_2 = 0^* 1 + q_2 1$ (Q, R, P)

$q_2 = 0^* 1 1^*$

$q_2 = 0^* 1 + q_1 q_2 q_3 1$

$q_2 = 0^* 1 + q_2 q_2 1 1$

$q_2 = 0^* 1 + q_2 1$

$R = QP^*$

$q_3 = q_2 \cdot 0$ ——— (2)

$q_2 = q_1 \cdot 1 + q_2 \cdot 1 + q_3 \cdot 1$

$q_2 = q_2 \cdot 1 + q_1 \cdot 1 + q_3 \cdot 1$

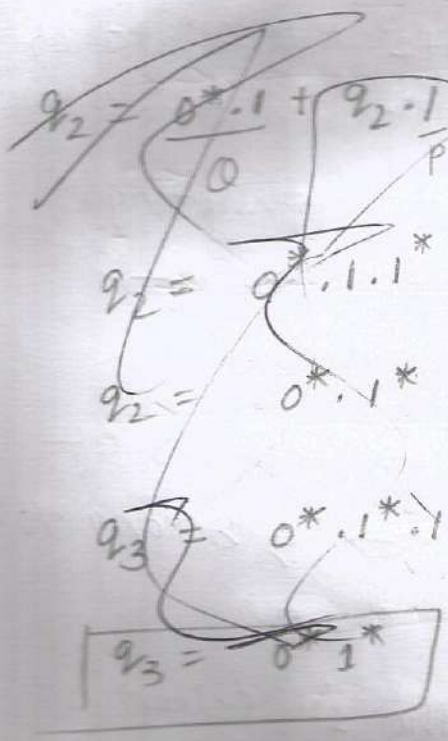
$q_2 = q_2 \cdot 1 + 0^* \cdot 1 + q_3 \cdot 1$ ——— (3)

$q_3 = q_2 \cdot 0$ ——— (4)

Put in q_3 in eqn (3)

$q_2 = q_2 \cdot 1 + q_3 \cdot 1 + 0^* \cdot 1$

$q_2 = q_2 \cdot 1 + q_2 \cdot 0 + 0^* \cdot 1$



$q_2 = (1+01)q_2 + 0^* \cdot 1$

R P R Q

P does not contain $\in$

$R = QP^*$

$q_2 = 0^* \cdot 1 \cdot (1+01)^*$

$q_2 = 0^* \cdot 1 \cdot (1+01)^*$

Putting in eqn (4)

$q_3 = q_2 \cdot 0$

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

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Equivalence to find the finite Automata comparison Method :-

Process :-

Let M and M' to be the finite Automata of Σ

We construct a Comparison table consisting of $(n+1)$ column where n is the no. of input symbols.

The first column consist of pairs of vertices of the form (q, q') where q belong to M and $q' \in M'$

if q, q' appears in some row of the 1st column in the corresponding entry A-column ($A \in \Sigma$) is q_A, q'_A where q_A, q'_A are reachable from q, q' respectively on application of A (i.e. by A-paths)

Here are two cases which specify whether M and M' are equivalent or not

Case 1 :- if we reach a pair q, q' s.t that q is the final state of M and q' is non-final state of M' or vise-varse q is non final and q' is the final we terminate the construction and conclude M and M' are not equivalent.

Case 2 :- Here the construction is terminated when No new element appears in the 2nd and the subsiquent column which are not in come first column (i.e. when all the elements in 2nd are subsiquent column appear in the 1st

In this case we conclude that M and M' are equivalent

Q. Consider the following two DFA's M, and M' over $\Sigma = \{c, d\}$ as given below determine whether M and M' are equivalent or not

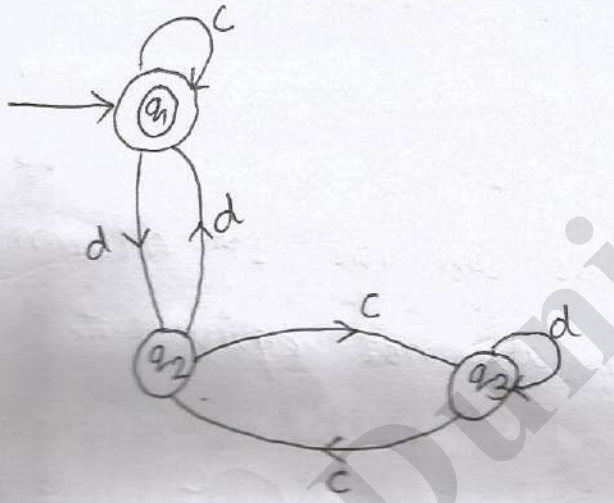


fig M

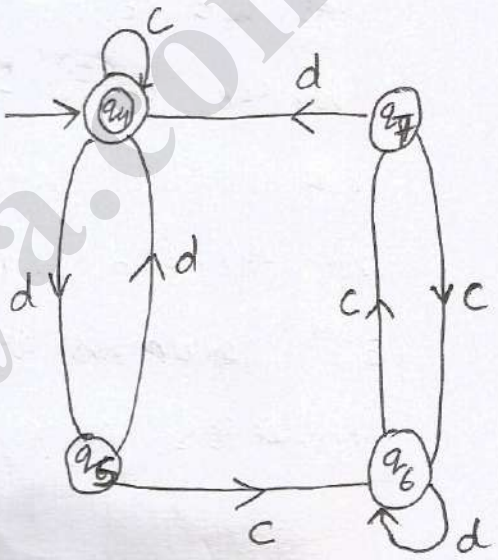


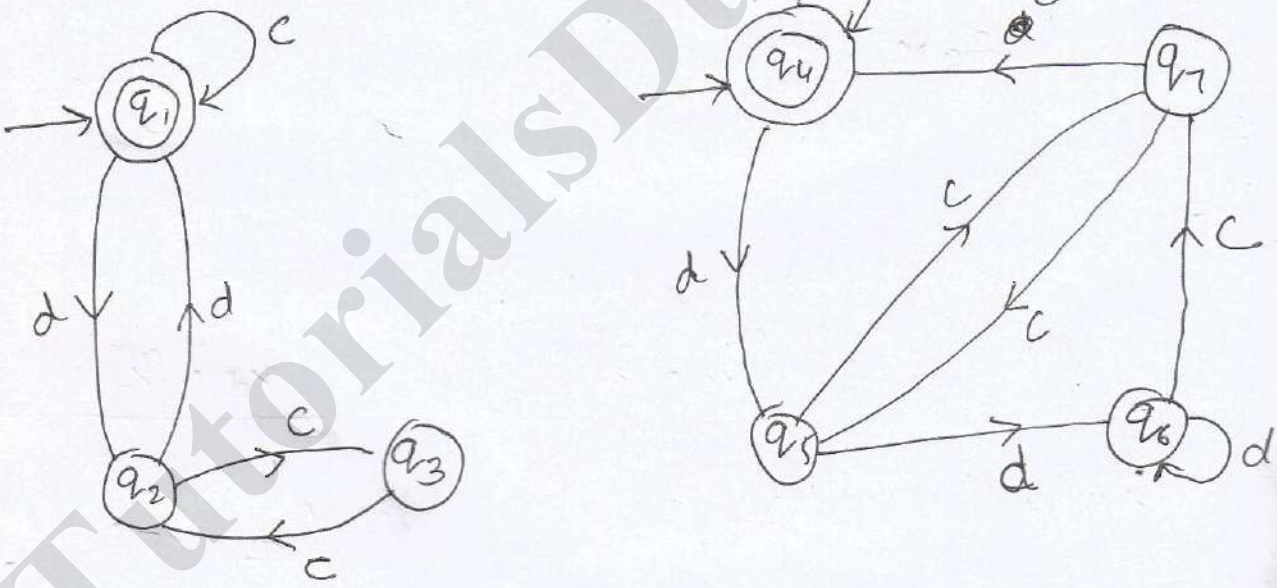
fig M'

Soln :-

(q, q')	(q_c, q'_c)	(q_d, q'_d)
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_6)	(q_1, q_4)
(q_3, q_6)	(q_2, q_7)	(q_3, q_6)
(q_2, q_7)	(q_3, q_6)	(q_1, q_4)

Description :-
initial state in M and M' are q_1 and q_4 respectively Hence the 1st element of the 1st column in the comparison table must be the 1st element in the 2nd column is (q_1, q_4) since both ~~q_1, q_1~~. q_1 and q_4 are ~~c~~ ~~neighbour~~ reacable As we do not get ~~copy~~ ~~here~~ a pair of (q, q') where q is the final state and q' is the non final state (or vise versa) at every row we proceed untill all the elements in the 2nd or 3rd columns are also in the 1st column
therefore M and M' are not equivalent.

Q.



(q, q')	(q_c, q_c')	(q_d, q_d')
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_6)	(q_1, q_6)

M and M' are not equivalent ∴

Q Is $L = a^{2n}$ $n \geq 1$ Regular language ?

a^{2n}

By pumping lemma theorem

Let L is RL

$z \in L$

$z = a^{2k}$

$uv^iw = a^{2k-1} \cdot a^i \cdot \varepsilon$

$a^{2k-1} \cdot a^i$

at $i = 1$

$a^{2k} \in L$

at $i > 1$

$a^{2k-1+i} \notin L$

at $i = 1$ L is RL

at $i > 1$ L is non Regular language

CFL

Context free language

language -	CFL
Corresponding Machine	PDA
Grammar	CFG/A

CFG :- A Grammar G is said to be CFG if every production is of the form $A \rightarrow \alpha$ where $A \in V_n$ and $\alpha \in (V_n \cup \Sigma)^*$

Derivation Trees :- Derivations in a CFG can be represented using trees such trees representing derivation are called derivations trees. derivation trees are also said to be parse tree

Assumption / properties :- i) every vertex has the level which is variabal or terminal or null.

ii) The root has level S

iii) The level of an internal vertex is variable

(iv) A vertex n is a leaf if it's level is $a \in \Sigma$ or NULL.

ex. Let $G = (\{S, a\}, \{a, b\}, \{P\}, \{S\})$

where P Consist of $S \rightarrow aAS \mid a \mid SS$

$A \rightarrow SbA \mid ba$

We have to Construct derivation tree for $w = aabaa$

$S \rightarrow aAS \mid a \mid SS$

$A \rightarrow SbA \mid ba$

~~$S \rightarrow SS$ aAS~~

~~aAS $aSbAS$ abA~~

~~ab~~ $aabbaa$

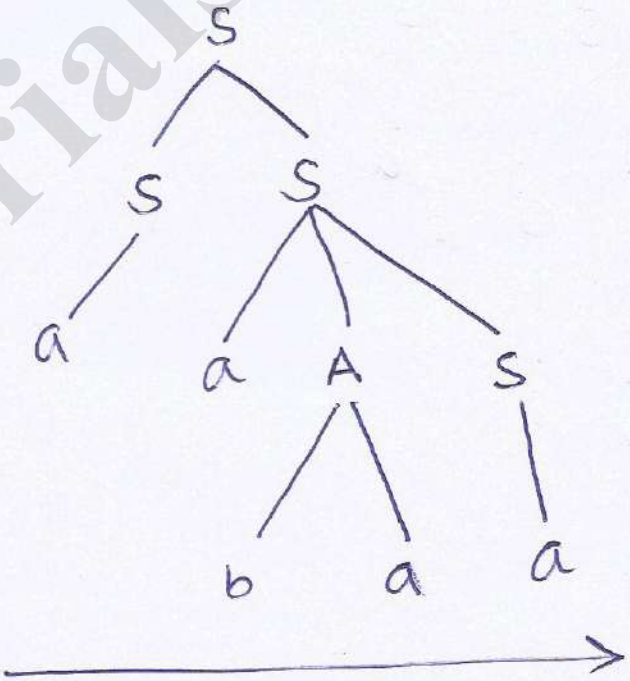
$S \rightarrow SS$

$S \rightarrow aS$ using $S \rightarrow a$

$S \rightarrow aaAS$ using $S \rightarrow aAS$

$S \rightarrow aabaS$ using $A \rightarrow ba$

$S \rightarrow aabaa$ using $S \rightarrow a$



it's a left most Derivation Trees

Two Types of Derivations Tree :-

i) Left Most Derivation Tree

ii) Right Most Derivation Tree

Right Most Derivation Tree's

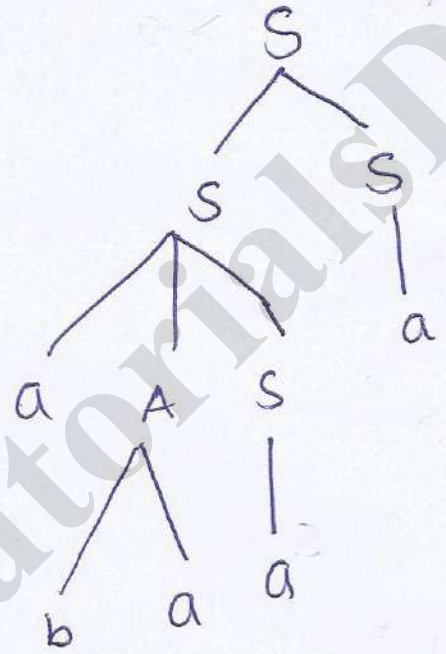
$S \rightarrow SS$

$S \rightarrow Sa$ using $S \rightarrow a$

$S \rightarrow aASa$ using $S \rightarrow aAS$

$S \rightarrow aAaa$ using $S \rightarrow a$

$S \rightarrow abaaa$ using $A \rightarrow ba$



Q.

Let G be the Grammar

$S \rightarrow 0B \mid 1A$

$A \rightarrow 0 \mid 0S \mid 1AA$

$B \rightarrow 1 \mid 1S \mid 0BB$

$w = 00110101$

a) LDT

b) RMT

a) $S \rightarrow 0\underline{B}$

$00\underline{B}B$ using $B \rightarrow 0BB$

$001\underline{S}B$ using $B \rightarrow 1S$

$0011\underline{A}B$ using $S \rightarrow 1A$

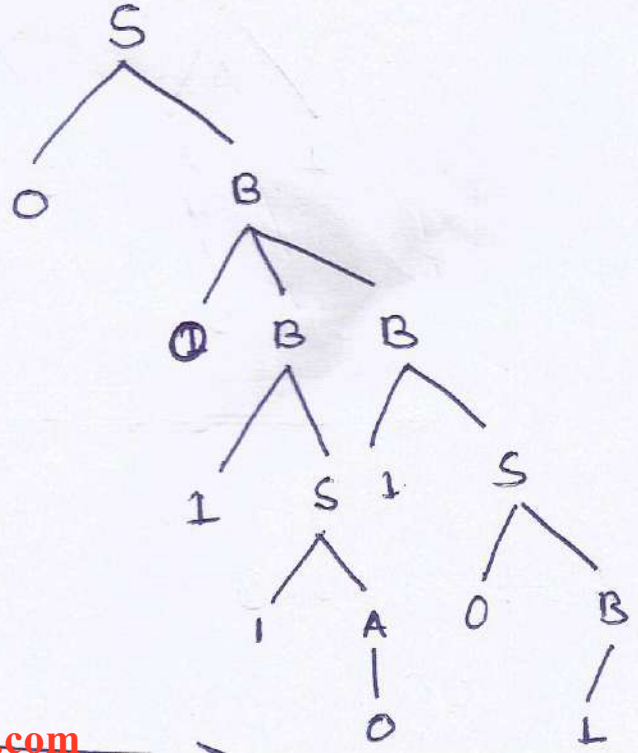
$00110B$

$00110\underline{1}S$ using $A \rightarrow 0$, $B \rightarrow 1S$

$0011010B$ using $S \rightarrow 0B$

00110101 using $B \rightarrow 1$

Left Derivation Tree



Right Most DT

b) $S \rightarrow OB$

$S \rightarrow OOBB$ using $B \rightarrow OBB$

$S \rightarrow OOB1S$ $B \rightarrow 1S$

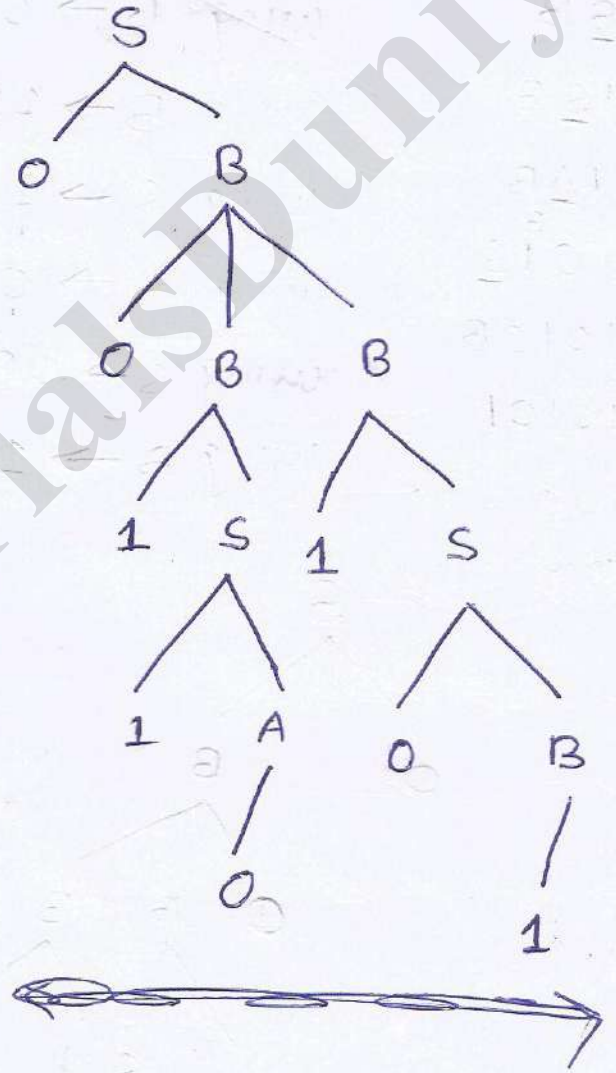
$S \rightarrow OOB10B$ $S \rightarrow OB$

$S \rightarrow OOB101$ $B \rightarrow 1$

$S \rightarrow OO1S101$ $B \rightarrow 1S$

$S \rightarrow OO11A101$ $S \rightarrow 1A$

$S \rightarrow 00110101$ $A \rightarrow 0$



00 110101

Ambuigrity in CFG (Ambigous Grammmar)

A Terminal string $w \in L(G)$ is Ambigous if there exist two or more derivation Trees for w (or there exist two or more left Most derivations of w)

ex. $G = (\{S\}, \{a, b, +, *\}, P, S)$

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

$w = a + a*b$

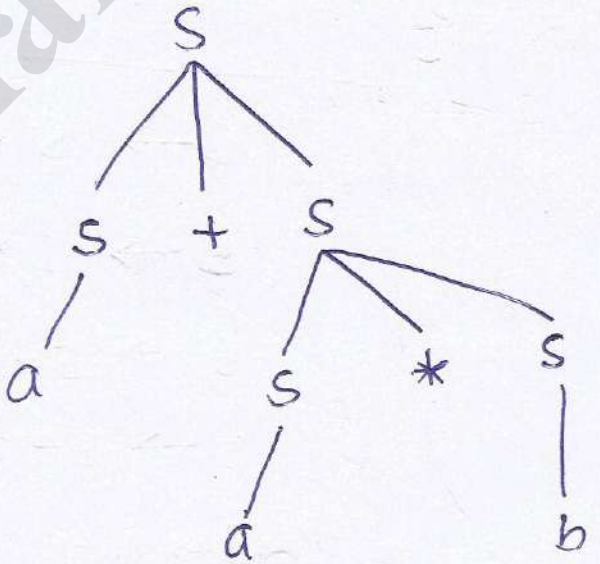
LDT:- $S \rightarrow S+S$

$a+S*S \quad S \rightarrow a$

$a+a*S \quad S \rightarrow a$

$a+a*b \quad S \rightarrow b$

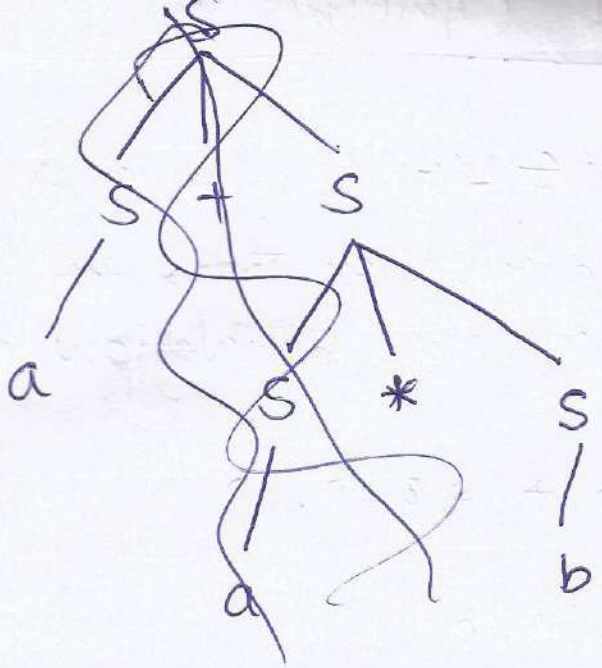
RD

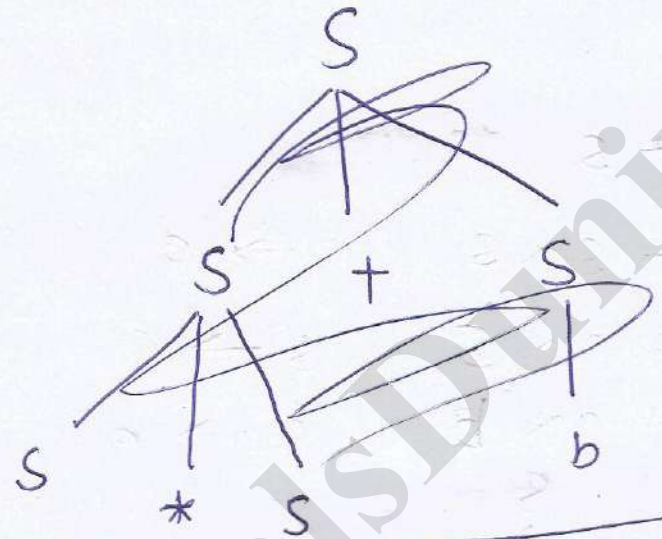


RDT:- $S \rightarrow S+S$

$S \rightarrow S+S*S$

using $S \rightarrow$





RDT

$S \rightarrow S * S$

$S \rightarrow S * b \quad (S \rightarrow b)$

$S \rightarrow S + S * b \quad (S \rightarrow S+S)$

$S \rightarrow S + a * b \quad (S \rightarrow a)$

$S \rightarrow a + a * b \quad (S \rightarrow a)$

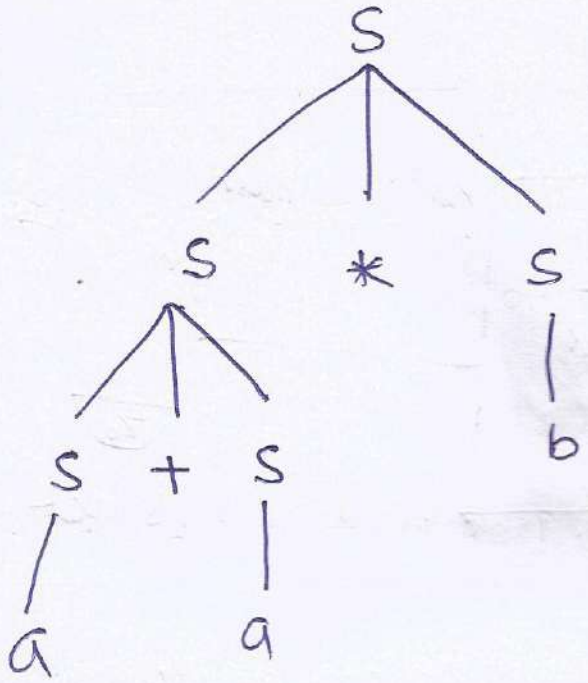
$S \rightarrow S * S$

$S * b$

$(S + S) * b$

$S + S * b$

$a + a * b$



PDA

PDA is a collection of 7 tuples

$M = Q, \Sigma, \Gamma, Z_0, q_0, q_f, \delta$ (Γ: Tape)

Where

Q = Set of all states

Σ = set of all input alphabets

Γ = Set of all Tape Symbols

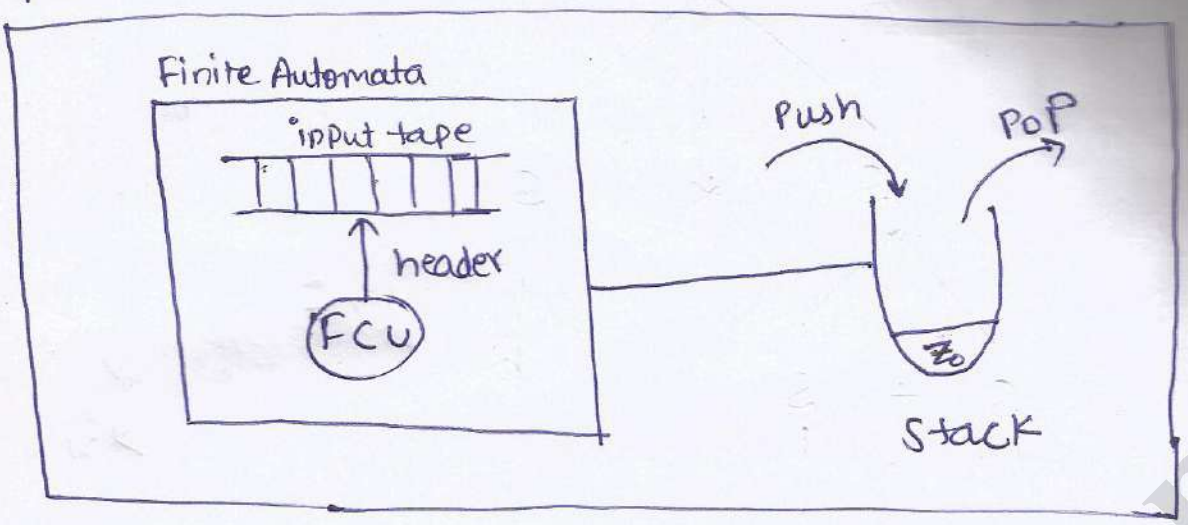
Z_0 = Top most symbol of stack

δ = $Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$

q_0 = initial state

q_f = final state

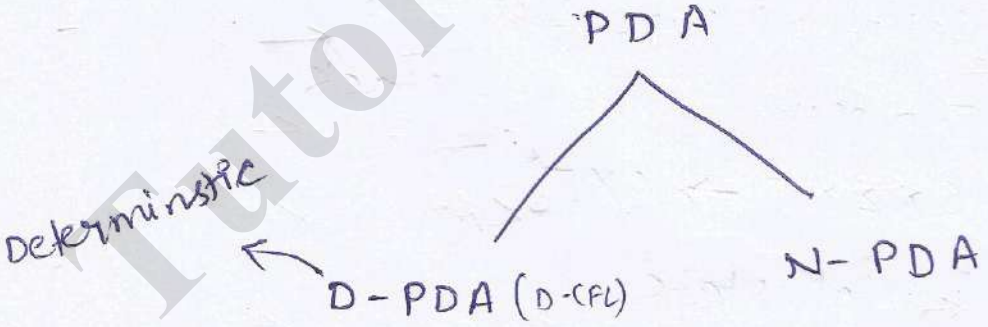
PDA



Structure of Push Down Automata

Points :-

1. PDA is more powerful than FA bcz of memory
2. The purpose of symbol Z_0 is to check that stack is empty or not
3. The language accepted by PDA is called CFL
4. PDA accept every Regular language and also accept sum of non-Regular language



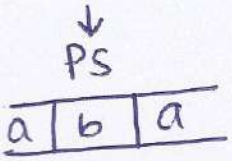
$$E(DPDA) \neq E(NPDA)$$

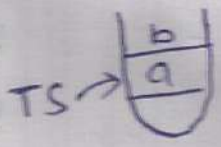
$$\therefore E(DPDA) < E(NPDA)$$

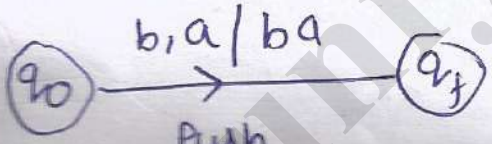
ID's for PDA :-

RE Decribe the movement of PDA. it Depends on 3 entities

1. Current state
2. Current processing symbol (P.S.)
3. Current top most symbol (T.S.)







b, a/a → skip

b, a/ε → Pop

ID's

$\delta(q_0, b, a) = (q_f, ba)$

- q_0 → Current state
- b → P.S.
- a → T.S.
- q_f → Next state
- ba → New T.S.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Block Diagram of PDA

Component of PDA :-

1. input tape
2. tape header
3. FCU (Finite control Unit)
4. Stack

Diagram :- PDA

By Default PDA is NPDA

Consider Gr use Productions are

Q. $S \rightarrow aAS/a$

$A \rightarrow SbA/ss/ba$

So that $S \rightarrow aabbaa$

Construct Possible derivations trees which yield aabbaa

Sol :-

$S \rightarrow aAS$

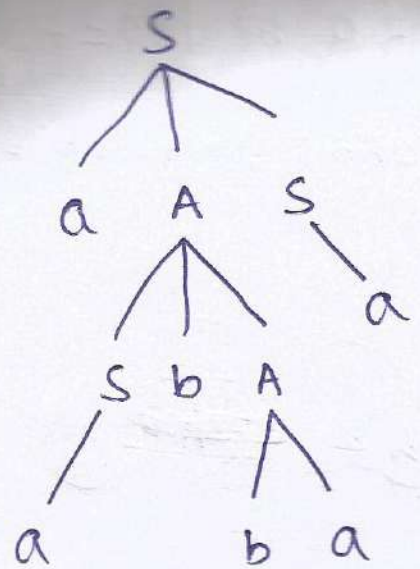
$S \rightarrow aSbAS$ using $A \rightarrow SbA$

$S \rightarrow aabAS$ using $S \rightarrow a$

$S \rightarrow aabbaS$ using $A \rightarrow ba$

$S \rightarrow aabbaa$ using $S \rightarrow a$

LMD Tree :-



RMD Tree :-

$S \rightarrow aAS$

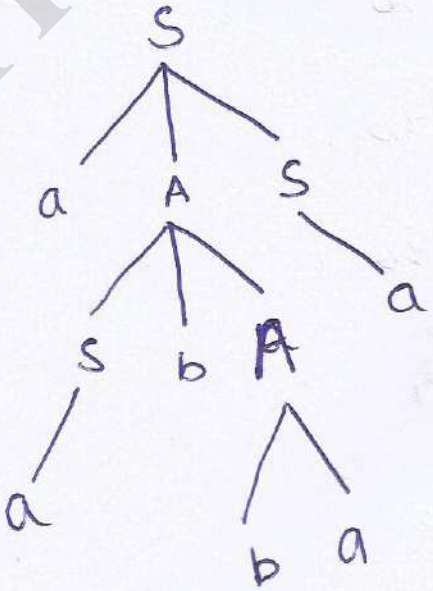
aAa using $S \rightarrow a$

~~$aSbAba$~~ using $A \rightarrow SbA$

$aSbAa$

$asbbaa$ using $A \rightarrow ba$

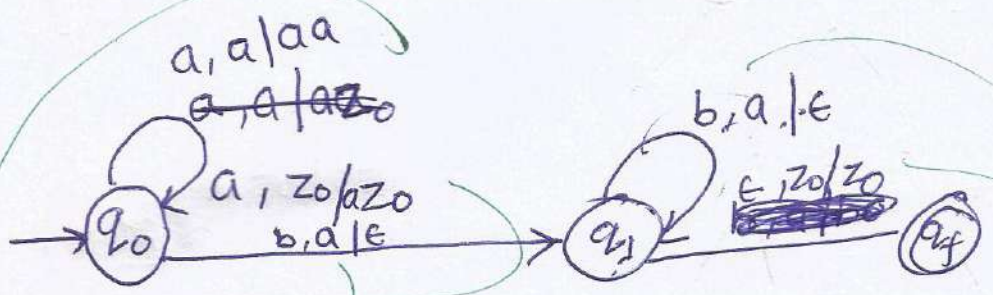
~~aS~~ $aabbaa$ using $S \rightarrow a$



Not Ambigious

Q. Construct PDA for $L = \{ a^n b^n \mid n \geq 1 \}$ check w=aabb for its acceptance?

Soln:- L = { ab, aabb, aaabbb ----- }



$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$

For check w = aabb :-

$aabb, Z_0, q_0$

abb, aZ_0, q_0

bb, aaZ_0, q_0

b, aZ_0, q_1

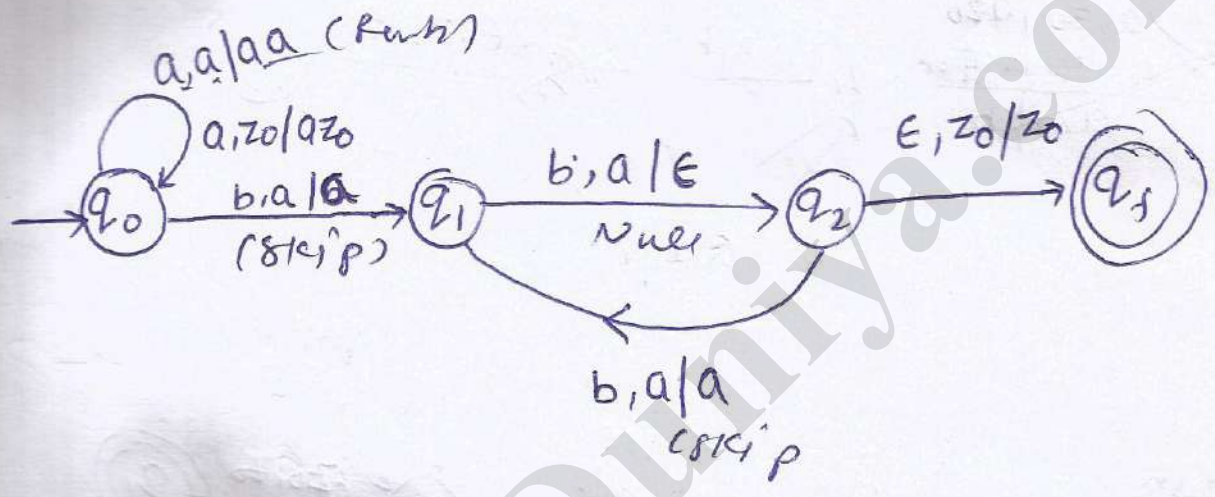
ϵ, Z_0, q_f

Accepted

Q. $L = \{a^n b^{2n} \mid n \geq 1\}$

Construct PDA ?

$L = \{abb, aabbbb, aaabbbbbb - - - - - - \}$



$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, a)$

$\delta(bq_1, b, a) = (q_2, \epsilon)$

$\delta(q_2, b, a) = (q_2, a)$

$\delta(q_1, b, a) \quad (q_2, \epsilon)$

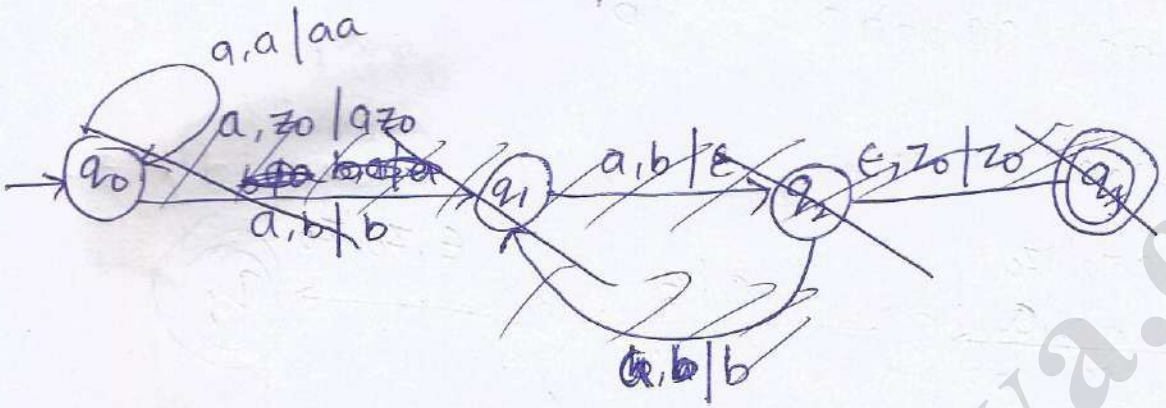
$\delta(q_2, \epsilon, z_0) = (q_f, z_0)$

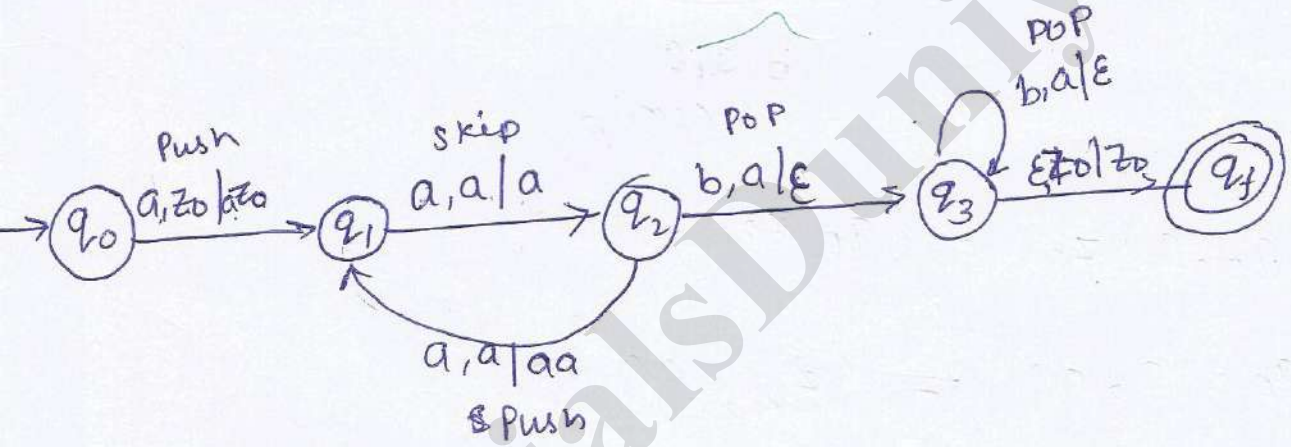
Ans

Construct PDA?

$L = \{a^{2n} b^n / n \geq 0 + 1\}$

$L = \{aab, aaaabb, aaaaaabbb, ----\}$





TD's :-

$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_1, a, a) = (q_2, a)$

$\delta(q_2, a, a) = (q_1, aa)$

$\delta(q_1, a, a) = (q_2, a)$

$\delta(q_2, b, a) = (q_3, \varepsilon)$

$\delta(q_3, \varepsilon, z_0) = (q_f, z_0)$

for check aaaaaabbb :-

aaaaaabbb Z_0 , q_0

aaaaabbb, aZ_0 , q_1 (Push)

aaaabbb, aaZ_0 , q_2 (Skip)

aaabbb, aaaZ_0 , q_1 (Push)

aabbb , aaaZ_0 , q_2 (Skip)

abbb , aaaaZ_0, q_1 (Push)

bbb , aaaaZ_0 , q_2 (Skip)

bb, aaaZ_0, q_3 (Pop)

b, aaZ_0, q_3 (Pop)

ϵ, Z_0, q_f (Pop)

Accepted

2. $L = \{a^m b^n \mid m < n\}$,

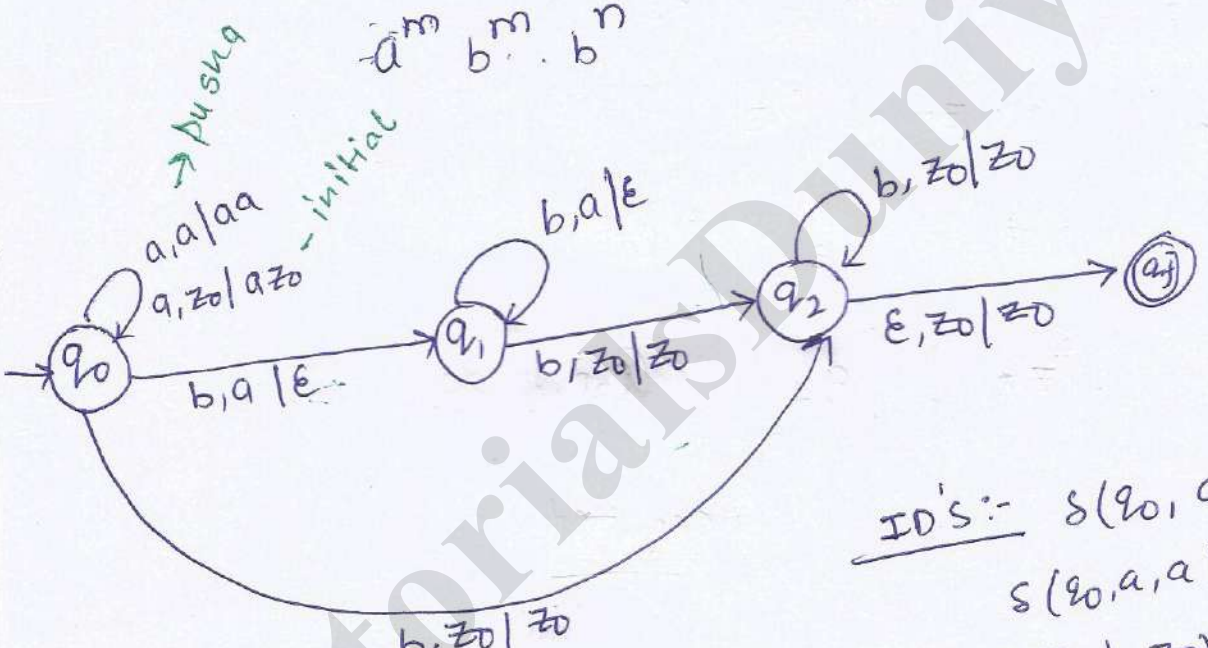
$m \geq 0$

$n \geq 1\}$ Construct PDA

~~$L = \{b \to abb, aabbb \to aaabbbb \ldots\}$~~

$L = \{ b, bb, bbb - \ldots$
$abb, abbb, abbbb - \ldots$
$aabbb, aaabbbb, aaaabbbbb \ldots \}$

$a^m b^m . b^n$



ID's :-

$\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, z_0) = (q_2, z_0)$
$\delta(q_0, b, a) = (q_1, \epsilon)$
$\delta(q_1, b, a) = (q_1, \epsilon)$
$\delta(q_1, b, z_0) = (q_2, z_0)$
$\delta(q_2, \epsilon, z_0) = (q_f, z_0)$

Check Acceptness :-

aaabbb , z_0 , q_0

aabbb , az_0 , q_0 (Push)

abbb , aaz_0 , q_0 (Push)

bbb , $aaaz_0$, q_1 (Pop)

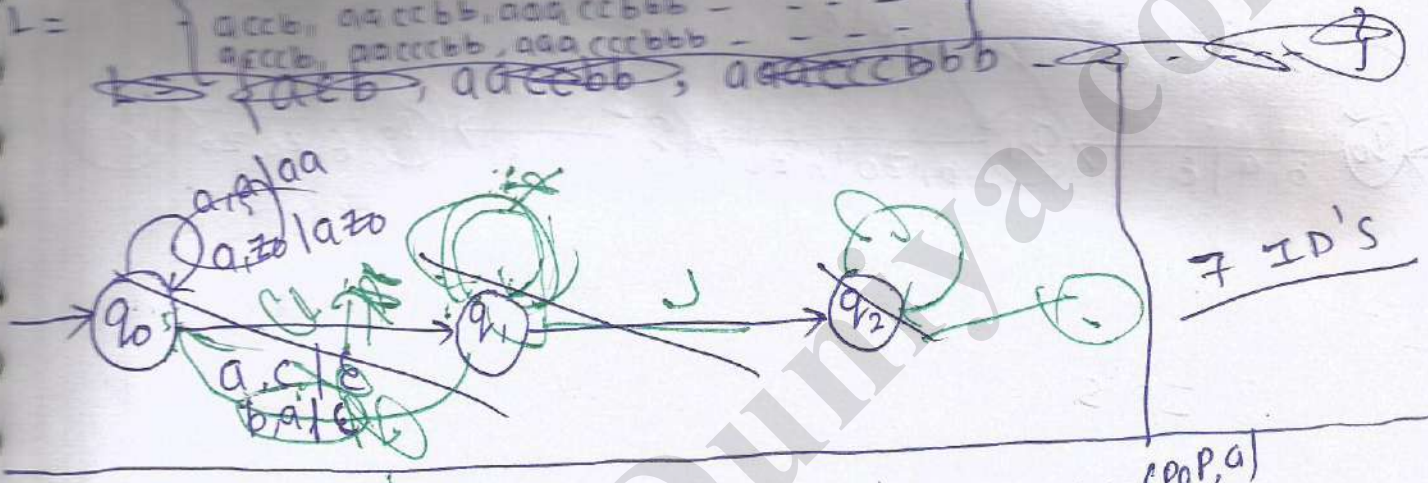
b , aZ_0 , q_1 (POP)

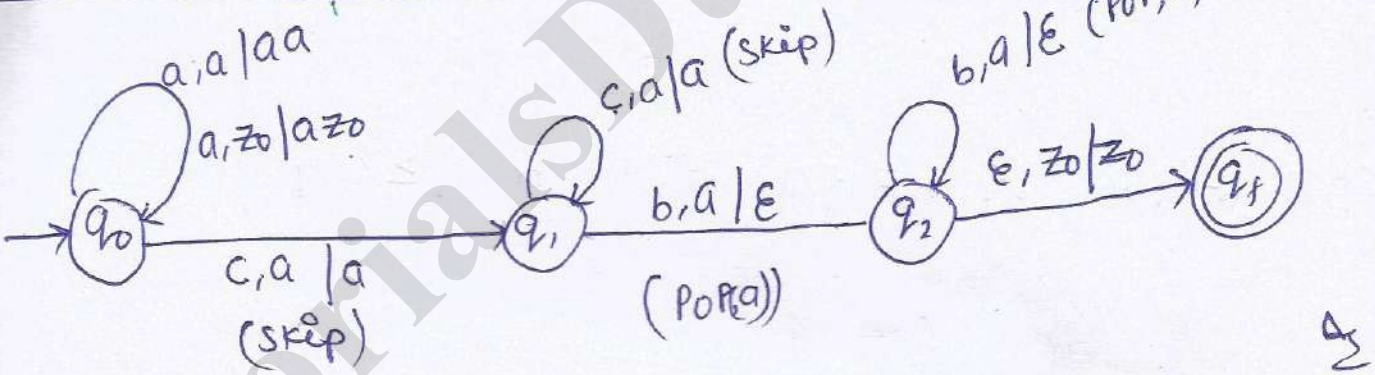
~~Z_0~~ ε , Z_0 , q_1 (POP)

Not accepted

Q $L = \{ a^n c^m b^n \mid n \geq 1, m \geq 1 \}$ construct PDA ?

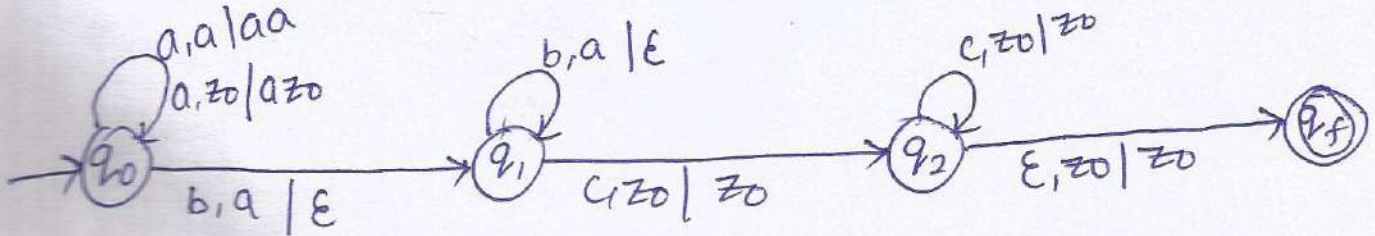
$L = \{ acb, aacbb, aaacbbb, \ldots \}$





Q $L = \{ a^m b^m c^n \mid m,n \geq 1 \}$

$L = \{ abcc, aabbccc, aaabbbcccc, \ldots \}$
$\{ abcc, aabbcc, aaabbbcc0, \ldots \}$
$\{ abc, aabbc@, aaabbbc, \ldots \}$

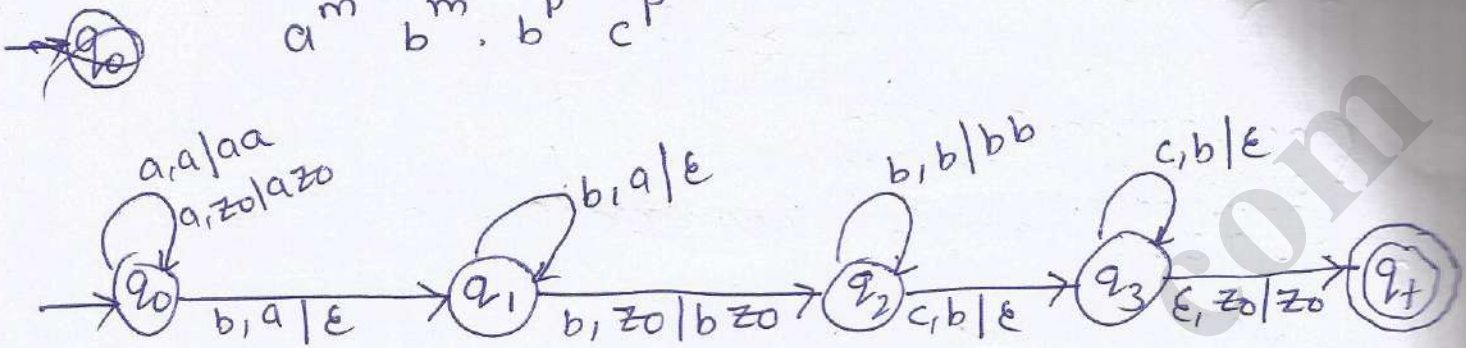


Q $L = \{a^m b^n c^p \mid n = m+p\}$

$m, n, p \geq 1$

$a^m \; b^{m+p} \; c^p$

$a^m \; b^m . b^p \; c^p$

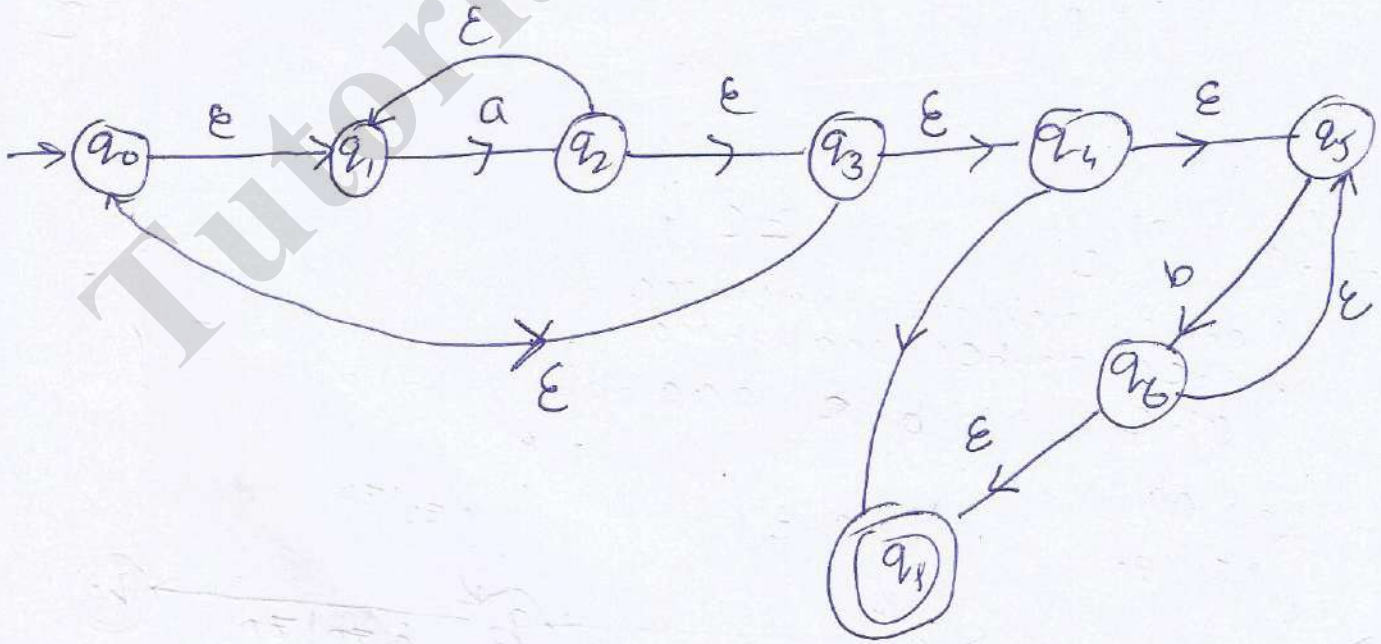


ID's :-

9 ID's

Imp.

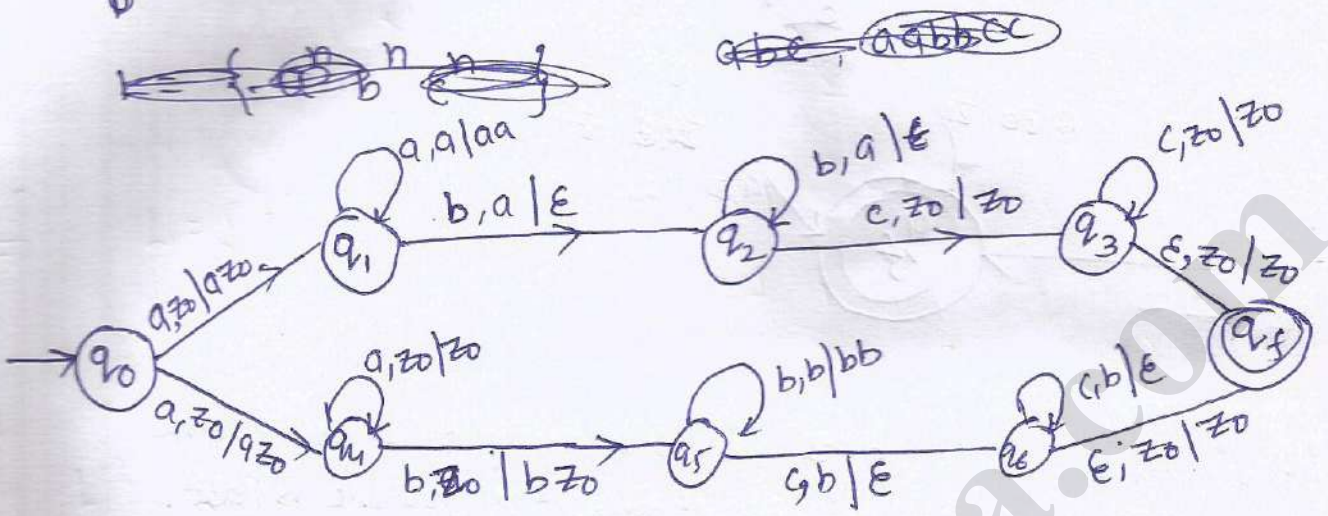
Q Find ε-NFA for $L = \{a^m b^n \mid m, n \geq 0\}$



Q $L = \{ a^m b^n c^p \mid m = n \text{ or } n = p \}$

$m n p \geq 1$

D



$L = \{ a^m b^n c^p \mid m = n \text{ or } mnp \geq 1 \}$

$a^m b^m c^p$

$\{ abc, aabbc, aaabbbc - - - - \}$

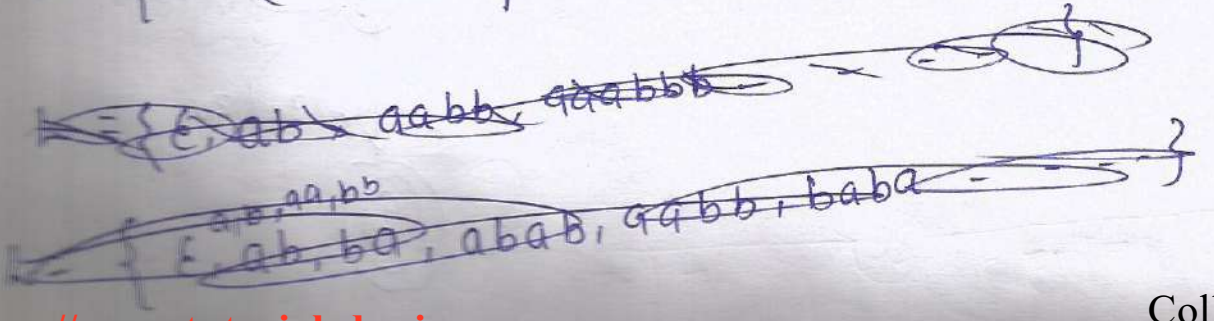
$L = \{ a^m b^n c^p \mid n = p \text{ or } mnp \geq 1 \}$

$a^m b^n c^p$

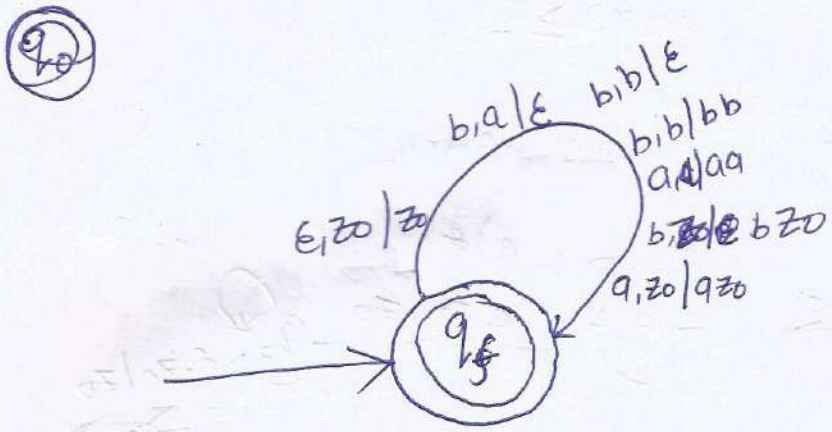
$L = \{ abc, abbcc, abbbccc - - - - \}$

it is NPDA bcz it has two out for the input from the same state. 14 ID's

Q $L = \{ w \in (a+b)^* \mid |a| = |b| \}$

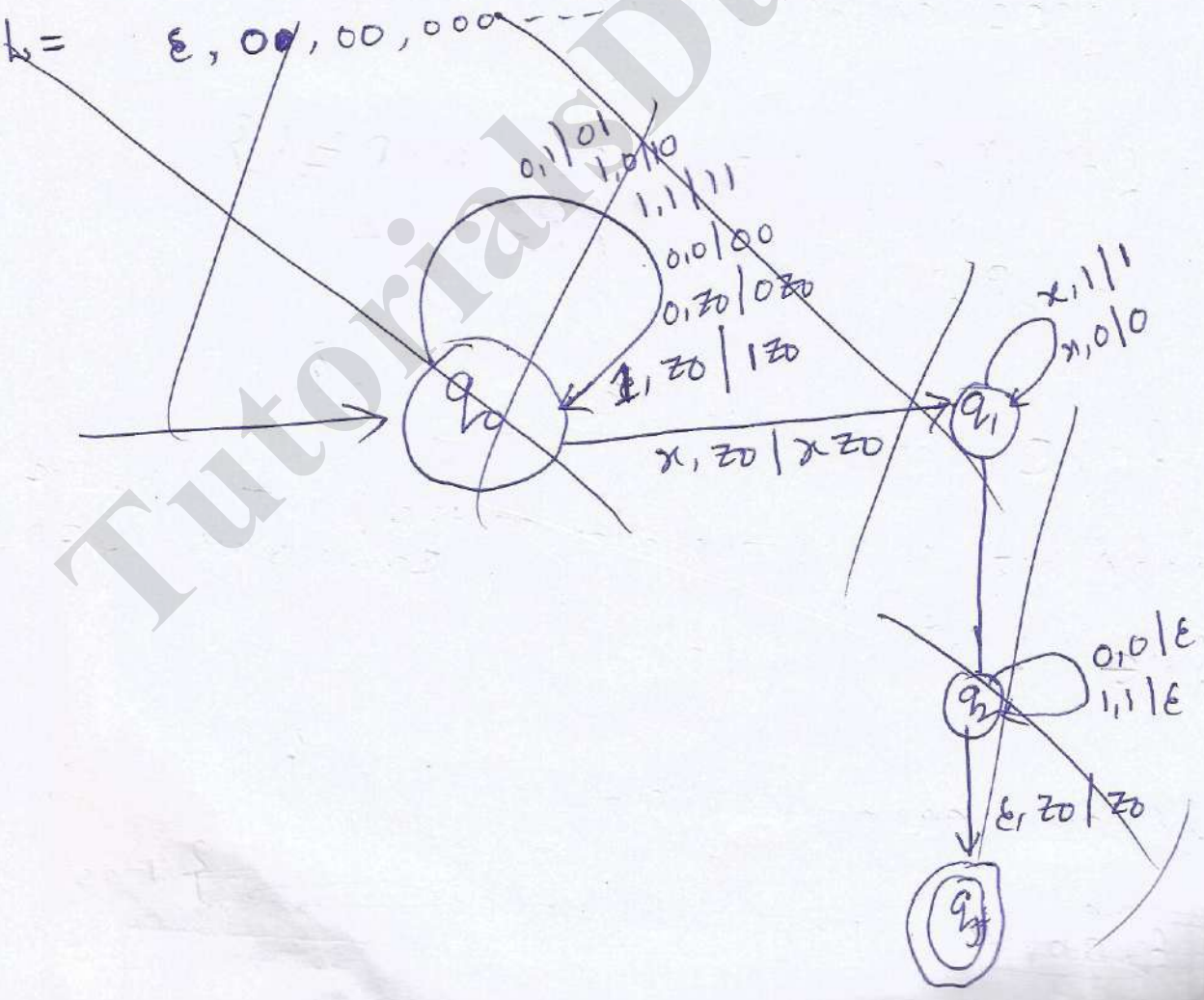


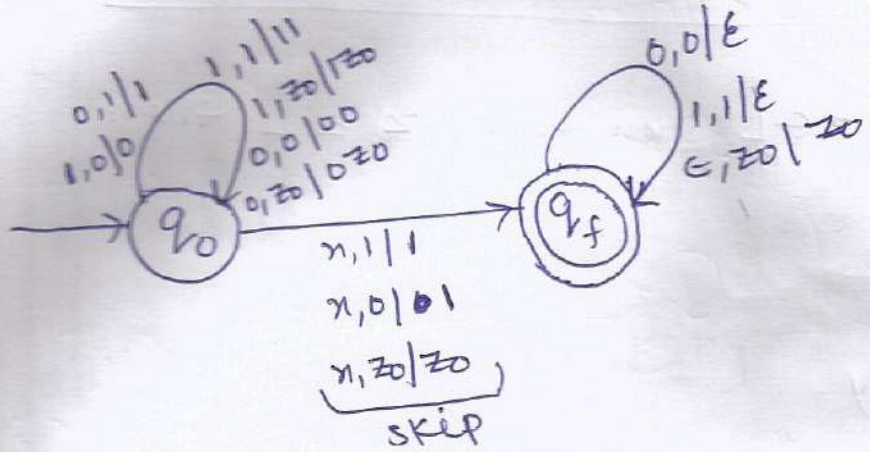
$L = \{ \varepsilon, ab, aabb, aaabbb, ------ \}$



it is also D-PDA bcz it has only and only one output for given input from the same state.

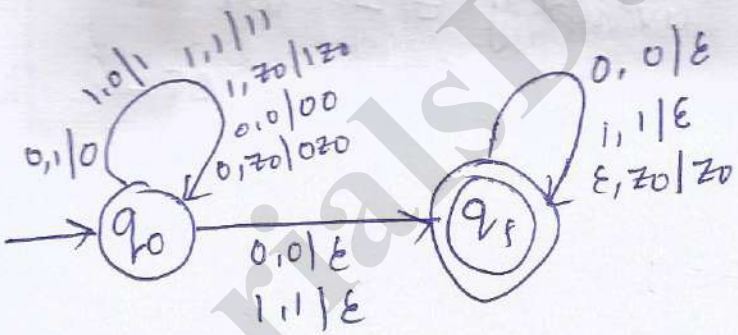
Q. $L = \{ wxw^R \mid w \in (0+1)^* \}$





is is also D-PDA

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



D-PDA Machine and Grammer CFL

Pumping Lemma for CFL

If L is any CFL and $z \in L$ such that $|z| \geq L$ then

$$u \cdot v^i \cdot w \cdot x^i \cdot y \in L$$

where $z = uvwxy$

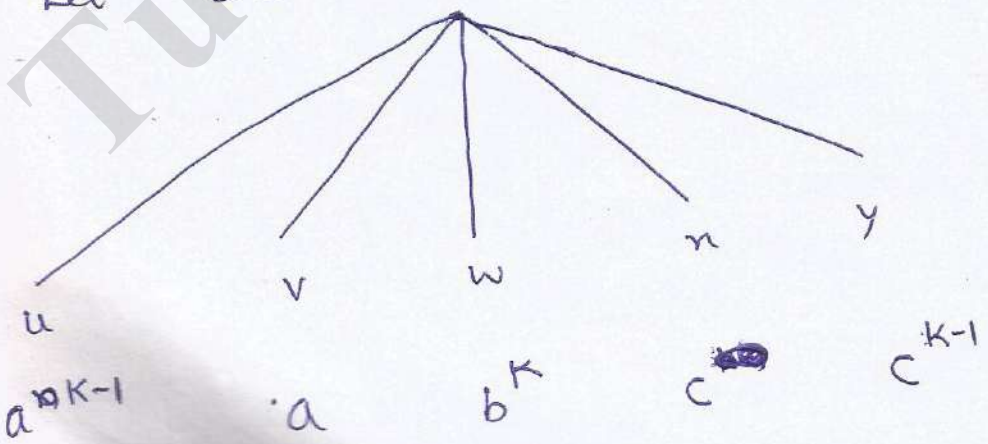
$$|uvx| \leq |z|$$

$$|vx| \neq 0$$

Pumping lemma for CFL is use to prove that sum of the language is NOT CFL. if there exist an integer i such that $uv^iwx^iy \notin L$ for at least one value of i then L is not CFL

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

Let $z = a^K b^K c^K$



TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

$uv^iwx^iy = a^{K-1} a^i b^K c^i c^{K-1}$

$=$ For $i = 2$

$a^{K-1} a^2 b^K c^2 c^{K-1}$

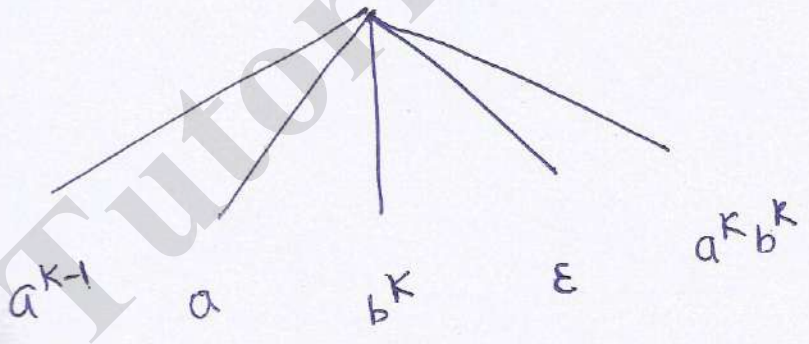
$a^{K+1} b^K c^{K+1} \notin L[\neq a^K b^K c^K]$

here at least one value which contain at least one value $i = 2$ then L is not CFL

Q. $L = \{ww \mid w \in (a+b)^+\}$ Language in CFL or not ?

Let $z = (a+b)^+ (a+b)^+$

Let $z = a^K b^K a^K b^K$



$uv^iwx^iy = a^{K-1} a^i b^K \varepsilon^i a^K b^K$

for $i = 2$

$a^{K+1} b^K a^K b^K$

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

Q $L = \{a^{2n} \mid n \geq 0\}$ chek CFL or not ?

Normal Forms for CFGs :-

1. **Chomsky normal form :-** A CFG is in Chomsky normal form if every production is of the form

$A \to a$ or $A \to BC$ or $S \to \wedge$

is in G if $\wedge \in L(G)$

be Assume that S does not appear R.H.S. on the production

Reduction to Chomsky Normal form :-

Step 1 :- elimination of null productions and Unit Productions

Step 2 :- elimination of terminals on R.H.S.

Step 3 :- (i) Restricting the no. of variables on R.H.S.

Q. Reduce the following Grammar G To CNF

G is $S \to aAD$

$A \to aB \mid bAB$

$B \to b$

$D \to d$

Sol^n Here there are no any NULL Production and ~~S → aabd~~ Unit Production then we can Proceed

Step2

Step2:- $G_1 = (V_N', \{a,b,d\}, P_1, S)$

where P_1 and V_N' are constructed as follow

(i) $B \to b$, $D \to d$ are included in P_1

(ii) $S \to aAD$ gives rise to $S \to CaAD$ and $Ca \to a$

$A \to aB$ gives rise to $A \to CaB$

$A \to bAB$ gives rise to $A \to CbAB$ and $Cb \to b$

$V_N' = \{S, A, B, D, Ca, Cb\}$

Step3:- P_1 consist of $S \to CaAD$, $A \to CaB \mid CbAB$, $B \to b$, $D \to d$, $Ca \to a$, $Cb \to b$

$A \to CaB$, $B \to b$, $D \to d$, $Ca \to a$, $Cb \to b$

are included into P_2

$S \to CaAD$ is replaced by $S \to CaC_1$ and $C_1 \to AD$

Similarly

$A \to C_b AB$ is replaced by $A \to C_b C_2$ where $C_2 \to AB$

Now $G_2 = (\{V_n''\}, \{a, b, d\}, \{P_2\}, \{S\})$

$V_n'' = \{S, A, B, D, C_a, C_b, C_1, C_2\}$

Where P_2 consist of

$A \to C_a B$

$B \to b$

$D \to d$

$C_a \to a$

$C_b \to b$

$S \to C_a C_1$

$A \to C_b C_2$

$C_1 \to AD$

$C_2 \to AB$

Q. $S \to a / AbB$

$A \to aA / a$

$B \to bB / b$

Covered into CNF

Step 1 :- Here there is no NULL and Unit Production we can Process to step 2

Step 2 :- $G_1 = (V_n', \{a, b\}, P_1, S)$

where P_1 and V_n' are constructed as follow

i) $S \to a$, $A \to a$, $B \to b$ are included in P_1

$S \to AbB$ gives rise to $S \to AC_bB$ and $C_b \to b$

$A \to aA$ gives rise to $A \to C_aA$ and $C_a \to a$

$B \to bB$ gives rise to $B \to C_bB$

$V_N' = \{S, A, B, C_a, C_b\}$

Step3 :- P_1 consist of $S \to a$, $A \to a$, $B \to b$

$S \to AC_bB$, $A \to C_aA$, $B \to C_bB$

$C_b \to b$, $C_a \to a$

$S \to a$, $A \to a$, $B \to b$, $A \to C_aA$, $B \to C_bB$

$C_b \to b$, $C_a \to a$ are included in P_2

$S \to AC_bB$ is replaced by $S \to AC_1$

where $C_1 \to C_bB$

Now $G_2 = (\{V_N''\}, \{a, b\}, P_2, S\}$

~~where P_2 and V_N'' are con~~

$V_N'' = \{S, A, B, C_a, C_b, C_1\}$

and P_2 consist of $S \to a$, $B \to C_bB$

$A \to a$, $C_a \to a$

$B \to b$, $C_b \to b$

$A \to C_aA$ $S \to AC_1$

$C_1 \to C_bB$

Q. $S \rightarrow \sim S \mid [S \supset) S] \mid P \mid q$

Soln There is no null and unit production, we can proceed to step 2

Let $G_1 = \{ \{V'_N\}, \{\sim, [, \supset,),], P, q\}, P, S\}$

where P_1 and V'_N are constructed as follows

i) $S \rightarrow P$, $S \rightarrow q$ are include in P_1

(ii) $S \rightarrow \sim S$, give rise to $S \rightarrow C_a S$, $C_a \rightarrow \sim$

$S \rightarrow [S \supset) S]$ give rise to $S \rightarrow C_b S C_c C_d S C_e$

where $C_b \rightarrow [$, $C_c \rightarrow \supset$, $C_d \rightarrow)$, $C_e \rightarrow]$

$V'_N = \{S, C_a, C_b, C_c, C_d, C_e\}$

P_1 consist of

$S \rightarrow C_a S \mid C_b S C_c C_d C_e \mid P \mid q$

$C_a \rightarrow \sim$, $C_b \rightarrow [$, $C_c \rightarrow \supset$, $C_d \rightarrow)$, $C_e \rightarrow]$

$S \rightarrow C_b S C_c C_d S C_e$ is replaced by $S \rightarrow C_b C_1$

where $C_1 \rightarrow S C_c C_d S C_e$

$C_1 \rightarrow S C_c C_d S C_e$ is replaced by $C_1 \rightarrow S C_2$

where $C_2 \rightarrow C_c C_d S C_e$

$C_2 \rightarrow C_c C_d S C_e$ is replaced by $C_2 \rightarrow C_c C_3$

where $C_3 \rightarrow C_d S C_e$

$C_3 \rightarrow C_d S C_e$ is replaced by $C_3 \rightarrow C_d C_4$

where $C_4 \rightarrow S C_e$

Now

$G_2 = (\{V_N''\}, \Sigma, P_2, S)$

V_N'' and P_2 are constructed as follow

$V_N'' = \{S, C_a, C_b, C_c, C_d, C_e, C_1, C_2, C_3, C_4\}$

P_2 consist of $S \to P|q|CaS|C_bC_1$

$C_a \to \sim$ $\quad C_c \to]$

$C_b \to [$ $\quad C_1 \to SC_2$

$C_c \to]$ $\quad C_2 \to C_cC_3$

$C_d \to)$ $\quad C_4 \to SC_c$

Q. $S \to aAbB$

$A \to aA|a$

$B \to bB|b$

Convert the Grammer into chomsky normal form

soln:-1. There is no null and Unit production we can Procede To step 2

2 $G_1 = (\{V_N'\}, \{a, b\}, P, S)$

where P_1 and V_N' are constructed as follow

(i) $A \to a$, $B \to b$ are included in P_1

$A \to aA$ give rise to $A \to C_aA$ and $C_a \to a$

$B \to bB$ give rise to $B \to C_bB$ where $C_b \to a$

$S \to aAbB$ give rise to $S \to C_aAC_bB$

$V_N = \{S, A, B, C_a, C_b\}$

$P_1 =$ $\quad A \to a|C_aA$ $\quad C_a \to a$

$B \to C_bB|b$ $\quad C_b \to b$

Step3 :- Production $A \to a \mid C_a A$

$B \to b \mid C_b B$

$C_a \to a$, $C_b \to b$

are included in P_2

$S \to C_a A C_b B$ is replaced by $S \to C_a C_1$ where $C_1 \to AC_b B$

$C_1 \to AC_b B$ is replaced by $C_1 \to AC_2$ where $C_2 \to C_b B$

$G_2 = (\{V_N''\}, \{a, b\}, P_2, S)$

where P_2 and V_N'' are constructed below

$P_2 =$ $A \to a \mid C_a A$

$B \to b \mid C_b B$

$S \to C_a C_1$

$C_2 \to C_b B$

$C_a \to a$

$C_b \to b$

$C_1 \to AC_2$

$V_N'' = \{S, A, B, C_a, C_b, C_1, C_2\}$

GNF (Greibach Normal Form)

CFG is in CNF if every production is of the form

$A \to a\alpha$ where $\alpha \in V_n$

$a \in \Sigma$ ($\alpha \xrightarrow[\text{may be}]{} \wedge$)

$\nexists$ $S \to \wedge$ is in grammar if $\wedge \in L(G)$

when $\wedge \in L(G)$ we assume that S does not appear on RHS of any production

Lemma (A) :- Let $G = (\{V_N\}, \Sigma, P, S)$ be a CFG let the set of A production be $A \to A\alpha_1 | A\alpha_2 | A\alpha_3 - -- A\alpha_n | \beta_1 | \beta_2 | \beta_3 - - - | \beta_n)$

[β_i does not start with A]

Let Z be a new variable here

$G_1 = (\{V_N'\}, \Sigma, P, S)$ $\nexists$ $V_N' = V_N \cup Z)$

where P_1 is define is as follows

(i) The set of A - Production in P_1 are

$A \to \beta_1 | \beta_2 | \beta_3 - - - | \beta_n$ with introduction of new variable

$A \to \beta_1 Z | \beta_2 Z | \beta_3 Z - - - - | \beta_n Z$

(ii) The set of Z production P_1 are

$Z \to \alpha_1 | \alpha_2 | - - - - \alpha_n$ with introduction of new variable Z

$Z \rightarrow \alpha_1 Z \mid \alpha_2 Z \mid \cdots \alpha_n Z$

(iii) The production for other variables are as S in P then G_1 is a CFG equivalent to G

Apply above lemma on Given CFG

$A \rightarrow aBD \mid bDB \mid C$

$A \rightarrow AB \mid AD$

$A \rightarrow \underbrace{aBD}_{\beta_1} \mid \underbrace{bDB}_{\beta_2} \mid \underbrace{C}_{\beta_3}$

$A \rightarrow \underbrace{AB}_{\alpha_1} / \underbrace{AD}_{\alpha_2}$

Let Z be a new variable

$A \rightarrow aBDZ \mid bDBZ \mid CZ$

$Z \rightarrow \alpha_1 \alpha_2$

$Z \rightarrow B/D$

$Z \rightarrow BZ/DZ$

Reduction to CNF :-

(i) Elimination of NULL and Unit production
(ii) Renaming of Production
(iii) Convert A_n — Production using lemma A
(iv) Modify A_i Production
(v) Modify Z_i Production
(vi) Construction of Grammar equivalent to given Grammar G.

Q. Construct the Grammar in GNF

$S \rightarrow AA \mid a$

$A \rightarrow SS \mid b$

Sol^n :- (i) The given grammar is in CNF S and A Renamed as A_1 and A_2 respectively

So the production are

$A_1 \rightarrow A_2 A_2 \mid a$

$A_2 \rightarrow A_1 A_1 \mid b$

As their is no unit and NULL Production So we procede to step 2

2

(i) A_1 Production are in required from. they are

$A_1 \rightarrow A_2 A_2 \mid a$

(ii) $A_2 \rightarrow b$ is in required form. Talking $A_2 \rightarrow A_1 A_1$ as the base Production using $A_1 \rightarrow A_2 A_2 \mid a$ in the base production.

then the resulting production are

$A_2 \rightarrow A_2 A_2 A_1$ and $A_2 \rightarrow aA_1$

then the input production are there

$A_2 \rightarrow b \mid aA_1 \mid A_2 A_2 A_4$

(iii) we have to apply lemma A to A_2 Production Introducing a new variable Z_2 to new produc

set of A_2 production in P_1 are

$A_2 \rightarrow \beta_1 \mid \beta_2$

$A_2 \rightarrow \beta_1 Z_2 \mid \beta_1 Z_2$

$A_2 \rightarrow b \mid aA_1$

$A_2 \rightarrow bZ_2 \mid aA_1Z_2$

set of Z_2 Production in P_1 are

$Z_2 \rightarrow \alpha$

$Z_2 \rightarrow \alpha Z_2$

$\therefore Z_2 \rightarrow A_2A_1$

$Z_2 \rightarrow A_2A_1Z_2$

4

(iv) (i) A_2 Production are

$A_2 \rightarrow aA_1 \mid b \mid aA_1Z_2 \mid bZ_2$

(ii) Among A_1 - Production we retain $A_1 \rightarrow a$ and replace all input production in $A_1 \rightarrow A_2A_2$

$A_1 \rightarrow aA_1A_2 \mid bA_2 \mid aA_1Z_2A_2 \mid bZ_2A_2$ —— (1)

The set of all modify Production are

$A_1 \rightarrow a \mid aA_1A_2 \mid bA_2 \mid aA_1Z_2A_2 \mid bZ_2A_2$ —— (2)

5. Z_2 Production to be modified are

$Z_2 \rightarrow A_2A_1 \mid A_2A_1Z_2$

Modified Z_2

$Z_2 \rightarrow aA_1A_1 \mid bA_1 \mid aA_1Z_2A_1 \mid bZ_2A_1$

$Z_2 \rightarrow aA_1A_1Z_2 \mid bA_1Z_2 \mid aA_1Z_2A_1Z_2 \mid bZ_2A_1Z_2$ —— (3)

Hence the equivelent Grammar in 'c'

$G' = (\{V_N'\}, \Sigma, P_1, A_1)$

$V_N' = \{A_1, A_2, Z_2\}$

P_1 consist of (1), (2) (3)

A_2

Q. Consider a CFG with the following question :-

$S \rightarrow ASA \mid B$

$B \rightarrow aCb \mid bCa$

$C \rightarrow ACA \mid A$

$A \rightarrow a \mid b$

i) Give three strings of length 7 in L(G)

ii) Are the following strings in L(G)

a) aaa b) bbb c) aba d) abb

(iii) is NULL(N) in L(G)?

Soln :- $S \rightarrow ASA$

~~$S \rightarrow AASAA$~~

~~$S \rightarrow AAASAAA$~~

~~$S \rightarrow aaa$~~

$S \rightarrow ABA$ [$S \rightarrow B$]

$S \rightarrow AbCaA$ [$B \rightarrow bCa$]

$S \rightarrow abACAaA$ [$C \rightarrow ACA$]

$S \rightarrow abaAaaa$ [$C \rightarrow a$]

$S \rightarrow abaaaaa$ [$A \rightarrow a$]

$S \rightarrow abaAaaa$

$S \rightarrow ababaaa$ [$A \rightarrow b$]

$S \rightarrow ASA$

$S \rightarrow ABA$ [$S \rightarrow B$]

$S \rightarrow AbCa$ [$B \rightarrow bCa$]

$S \rightarrow AbACAa$ [$C \rightarrow ACA$]

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

$S \rightarrow ASA$

$S \rightarrow aSA$ $(A \rightarrow a)$

$S \rightarrow aASAA$ $(S \rightarrow ASA)$

$S \rightarrow aaSAA$ $(A \rightarrow a)$

$S \rightarrow aaBAA$ $(S \rightarrow b)$

$S \rightarrow aaacbAA$ $(B \rightarrow acb)$

$S \rightarrow aaaAbAA$ $(C \rightarrow A)$

$S \rightarrow aaaabAA$ $(A \rightarrow a)$

$S \rightarrow aaaabaa$ $(A \rightarrow a)$

$S \rightarrow aaaabaa$ $(A \rightarrow a)$

a) aaa Not matching

b) bbb Not match

c) aba Not match

d) abb matched

$S \rightarrow B$

$S \rightarrow acb$

$S \rightarrow aAb$

$S \rightarrow aab$

NULL Production

Q. Construct PDA Aexcepting the set of all string over a, b with equal no. of A's and B's

L = {ab, aabb, aaabbb, aaaabbbb - - - - - }

ba, bbaa, bbbaaa, bbbb, aaaa - - - - -

$L = \{ (a+b)^* \mid |w|_a = |w|_b \}$

→ b ते a को null से

Push Down Automata and Context Free Grammer

Rule 1 :- $\delta(q, \wedge, A) = \{q, \alpha \mid A \to \alpha \text{ is in } P\}$

Rule 2 :- $\delta(q, a, a) = \{q, \wedge\}$ for every a in Σ

Question: Construct a PDA A equivalent is the following

CFG :- CFG is

$S \to OBB$

$B \to OS \mid 1S \mid 0$

Check whether 010^4 is in $N(A)$

Soln :- Define PDA as follow

$A = (\{q\}, \{S, B, 01\}, \{0, 1\}, \delta, q, S\$)$

$A = \boxed{((q)) \in, \quad V_N \cup \Sigma, \Sigma, \delta, q, S, \phi)}$

This is the general syntax for PDA

δ is define the following rule

R_1 $\delta(q, \wedge, S) = \{q, OBB\}$

R_2 $\delta(q, \wedge, B) = \{q, OS\}\{q, 1S\}\{q, 0\}$

R_3 $\delta(q, 0, 0) = \{q, \wedge\}$

R_4 $\delta(q, 1, 1) = \{q, \wedge\}$

$(q, 010^4, S)$

$\vdash (q, 010^4, OBB)$ By R_1

This symbol is used for check acceptency in CFG

$\vdash (q, 010^4, 01SB)$ By R_2

$\vdash (q, 010^4, 010BBB)$ By R_1

$\vdash (q, 010^4, 0100BB)$ By R_2

$\vdash (q, 010^4, 01000B)$ By R_2

$\vdash (q, 010^4, 010000)$ By R_2

$\vdash (q, 010^4, 010^4)$ B R_4

$\vdash (q, \wedge)$

$(q, 010^4, S)$

$\vdash (q, 010^4, OBB)$ By R_1

$\vdash (q, 10^4, BB)$ By R_3

$\vdash (q, 10^4, 1SB)$ By R_2

$\vdash (q, 0^4, SB)$ By R_4

$\vdash (q, 0000, OBBB)$ By R_1

$\vdash (q, 000, BBB)$ By R_3

$\vdash (q, 000, 000)$ By R_2

$\vdash (q, \wedge)$ By R_3

Q. $S \rightarrow aSb / A$

$A \rightarrow bSa \mid S \mid \wedge$

whether check acceptency "aabbbab"

Soln :-

R_1 $\delta(q, \wedge, S) = \{q, aSb\}\ \{q, A\}$

R_2 $\delta(q, \wedge, A) = \{q, bSa\}\ \{q, S\}, \{q, \wedge\}$

R_3 $\delta(q, a, a) = \{q, \wedge\}$

R_4 $\delta(q, b, b) = \{q, \wedge\}$

Acceptency check :-

$(q, aabbbab, S)$

$\vdash$ $(q, aabbbab, aSb)$ By R_1

$\vdash$ $(q, abbbab, @Sb)$ By R_3

$\vdash$ $(q, abbbab, aSbb)$ By R_1

$\vdash$ $(q, @bbbab, Sbb)$ By R_3

$\vdash$ $(q, bbbab, Abb)$ By R_1

$\vdash$ $(q, bbbab, bSab)$ By R_2

$\vdash$ $(q, bbab, Sab)$ By R_4

Aab

Q. $S \to ASA \mid bA$

$A \to B \mid S$

$B \to c$

$S \to ASA \mid bA$

$A \to c \mid ASA \mid bA$

$B \to c$

Soln :- Here, there are NO NULL production and Unit Production are then we can process step 2 →

$A \to B$

$A \to S$

$A \to c$

$A \to ASA \mid bA$

now there is no unit production

Step 2 :- $G_1 = (V_N', \{b, c\}, P_1, S)$

where P_1 and V_N' constructed as follow

(i) $B \to c$, $A \to c$ are included in P_1

(ii) $S \to bA$ gives rise to $S \to C_bA$ where $C_b \to b$

$A \to bA$ gives rise to $A \to C_bA$

$V_N' = \{S, A, B, C_b\}$

$P_1 : \quad S \to ASA \mid C_bA \quad , \quad C_b \to b$

$A \to c \mid ASA \mid C_bA$

$B \to c$

step3 :- $S \to C_bA$, $A \to c \mid C_bA$ and $B \to c$

$C_b \to b$ are included in P_2

$S \to ASA$ is replaced by $S \to AC_1$ where

$C_1 \to SA$

$A \to ASA$ is replaced by $A \to AC_2$ where

$C_2 \to SA$

Now $G_2 = (\{V_N''\}, (b,c), P_2, S)$

Where P_2 and V_N'' are constructed below

$V_N'' = \{S, A, B, C_b, C_1, C_2\}$

P_2 : $S \to AC_1 \mid C_bA$, $C_1 \to SA$

$A \to c \mid AC_1 \mid C_bA$, $C_b \to b$

$B \to c$, $C_2 \to SA$

Ans

1. Find Equivalent Grammar and Equivalent PDA to the given language

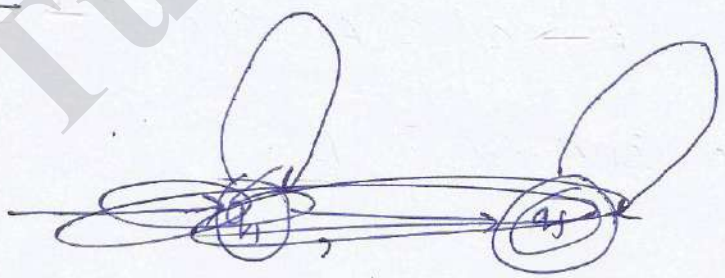
$L = \{ww^R \mid w \in (a+b)^*\}$

$S \to aSA \mid bSB \mid aA \mid bB$

$A \to a$

$B \to b$

PDA :-



R_1 : $\delta(q, \wedge, S) = (q, aSA \mid bSB \mid aA \mid bB)$

R_2 : $\delta(q, \wedge, A) = (q, a)$

R_3 : $\delta(q, \wedge, B) = (q, b)$

$R_4 \quad (q, a, a) = (q, \wedge)$

$R_5 \quad (q, b, b) = (q, \wedge)$

Q. find the RE for the set of L string in which every 0's followed by at least 2 one's

ε, 1, 011, 0111,

~~RE~~ ~~(1+011)*~~

$W \in L$

$1^* + (011)^*$

$(1^* + (011)^*)^*$

$(1 + 011)^*$

Q. no. of string of length ≤ 3 generated by RE $(0+10)^*$

$L = \{\varepsilon, 010, 001, 000, 011, 0, 10, 00, 100\}$

$L_0 = \varepsilon$

$L_1 = 0$

$L_2 = 10, 00$

$L_3 = 100, 000, 010$

Q. Which of the following RE does not contain the sub string 1101

a) $(10)^* 010^*$ NO → 10101001010 [0

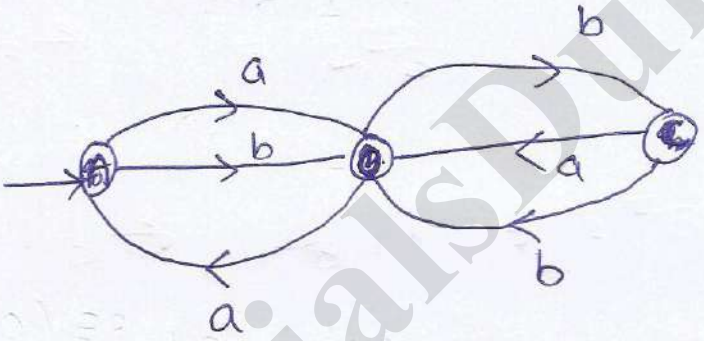
b) $01^* 01^*$ = {0 1101}, {0 1101}

c) $(11)^* 0 (01)^*$ NO

d) $(1+0)^*$

Ans a and c

Q find RE ?



$R = \emptyset P^*$

Arden's theorem :-

$A = B^a + \varepsilon$

$A = \varepsilon + Ba$

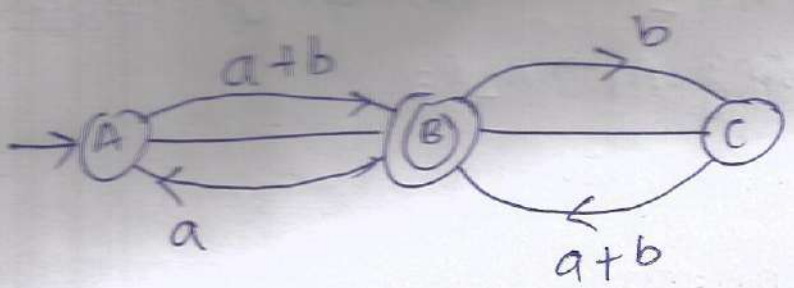
R Q P

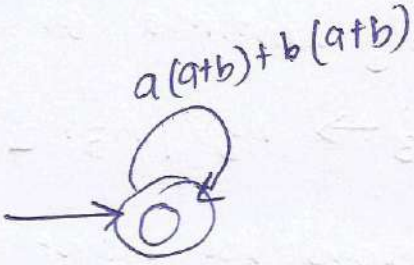
$A = \varepsilon a^*$

$A = a^*$

$B = Aa + Ab$

$B = A(a+b)$





$$a(a+b) + b(a+b)$$

$$r = [a(a+b) + b(a+b)]^*$$

$$R = [(a+b)^2]^*$$

Q. Convert the Given Grammar into GNF

$S \rightarrow AB$

$A \rightarrow BS \mid b$

$B \rightarrow SA \mid a$

Sol^n :- The Given Grammar is in CNF S, A and B are renaming as A_1, A_2, A_3 respectively, so the Production are -

$A_1 \rightarrow A_2A_3$ $\qquad$ $A_3 \rightarrow A_1A_2 \mid a$

$A_2 \rightarrow A_3A_1 \mid b$

As their is no NULL or Unit Production so we can Proceeds to step 2

Step 2:- (i) $A_1 \rightarrow A_2A_3$ is in required form.

(ii) $A_2 \rightarrow b$ is in required form

(iii) $A_2 \rightarrow A_3A_1 \mid b$ are in required form

(iii) $A_3 \rightarrow a$ is in required form

Apply lemma to $A_3 \rightarrow A_1 A_2$. The Resulting Productions are $A_3 \rightarrow A_2 A_3 A_2$ using $A_1 \rightarrow A_2 A_3$

Apply lemma on A_3 Again

$A_3 \rightarrow A_3 A_1 A_3 A_2 \mid b A_3 A_2$

Step3 Now we have to apply lemma A on A_3 Prod^n introducing a new variable Z_3

$A_3 \rightarrow \underbrace{a}_{\beta_1} \mid \underbrace{b A_3 A_2}_{\beta_2} \mid \underbrace{A_3 A_1 A_3 A_2}_{\alpha}$

$A_3 \rightarrow a Z_3 \mid b A_3 A_2 Z_3$

$Z_3 \rightarrow A_1 A_3 A_2$

$Z_3 \rightarrow A_1 A_3 A_2 Z_3$

Now A_3 - Production are

$A_3 \rightarrow a Z_3 \mid b A_3 A_2 Z_3 \mid a \mid b A_3 A_2$ —— (1)

Step 4 Among A_2 Production we retain $A_2 \rightarrow b$ and replace all the input production in

$A_2 \rightarrow A_3 A_1$

$A_2 \rightarrow a A_1 \mid b A_3 A_2 A_1 \mid a Z_3 A_1 \mid b A_3 A_2 Z_3 A_1$

The set of all modify production are

$A_2 \rightarrow b \mid a A_1 \mid b A_3 A_2 A_1 \mid a Z_3 A_1 \mid b A_3 A_2 Z_3 A_1$ —— (2)

For A_1 Production we replace the A_2 in $A_1 \rightarrow A_2 A_3$

$A_1 \rightarrow b A_3 \mid a Z_3 A_1 A_3 \mid b A_3 A_2 Z_3 A_1 A_3 \mid a A_1 A_3 \mid b A_3 A_2 A_1 A_3$ —— (3)

Step 5 :- Z_3 Production to be modify are

$$Z_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2Z_3$$

$$Z_3 \rightarrow bA_3A_3A_2 \mid aZ_3A_1A_3A_3A_2 \mid aA_1A_3A_3A_2 \mid$$
$$bA_3A_2Z_3A_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2$$

$$Z_3 \rightarrow bA_3A_3A_2Z_3 \mid aZ_3A_1A_3A_3A_2Z_3 \mid aA_1A_3A_3A_2Z_3$$
$$bA_3A_2Z_3A_1A_3A_3A_2Z_3 \mid bA_3A_2A_1A_3A_3A_2Z_3$$

Q. Find a Grammar in GNF equivalent to the Grammar A_3

$$E \rightarrow E + T / T$$
$$T \rightarrow T \times F / F$$
$$F \rightarrow (E) / a$$

Clower properties of D-CFL :-

Closed -

i) Compliment
ii) intersection in regular set
iii) quotient with regular set

Not Closed -

(i) Union
(ii) intersection
(iii) Concentation
(iv) Klean clouser

v

Clower properties of CFL :-

Closed :-

(i) Union
(ii) Concentation
(iii) Klean clouser
(iv) Reversal
(v) quotient with R set
(vi) intersection with R set

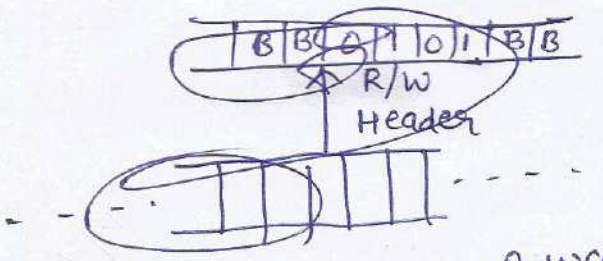
Not closed :-

(i) intersection (ii) Compliment

Turing Machine

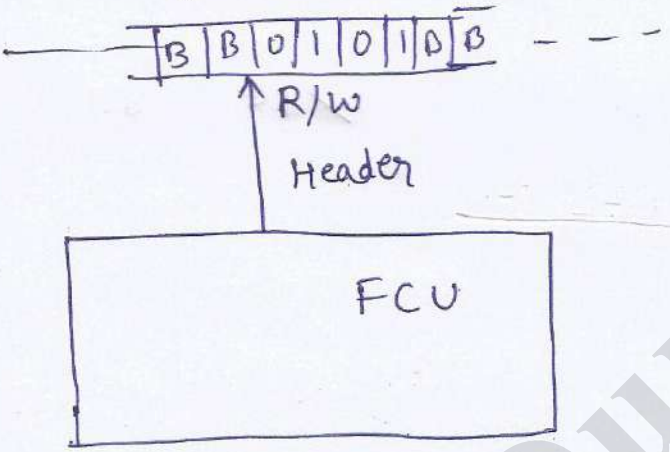
11-Sep-2018

Block diagram of Turing machine :-

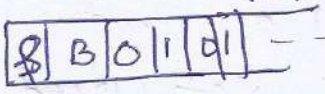


2 way

multitape (infinite tape)



one way infinite tape



(1A)

3 components in Turing Machine :-

(i) two way infinite tape

(ii) Read/write header

(iii) Finite control unit

infinite tape :-

Read/write Header :- Input टेप।

TM is . DTM when by default

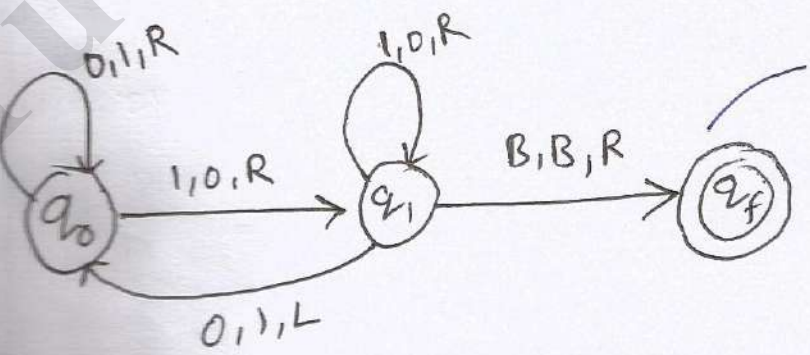
TM is accepted two language
i) Recursive
ii) Recursive innumerable language

Detination :- TM is collection of 7 tuples

$$M = \{ Q, \Sigma, \Gamma, B, \delta, q_0, F \}$$

where
- Q set of state
- Σ input symbols
- Γ ~~top most symbol~~ set of all tap Symbol
- B Blank symbol
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{ L, R \}$
- q_0 initial state
- f final state

ex.



Transition diagram

$\delta(q_0, 0) = (q_0, 1, R)$

$\delta(q_i, a) = (q_j, X, R)$
or
(q_j, X, L)

Representation of Turing Machine :-

it can be represented by two ways-

i) Transition table

(ii) Transaction diagram

Σ	q_0	q_1	q_f
0	q_0	q_0	
1	q_0	q_1	

	0	1	B
q_0	$(q_0,1,R)$	$(q_1,0,R)$	halt
q_1	$(q_0,1,L)$	$(q_1,0,R)$	(q_f,B,R)
q_f	halt	halt	halt

ID's

$\delta(q_0,0) = (q_0,1,R)$

$\delta(q_0,1) = (q_1,0,R)$

$\delta(q_1,0) = (q_0,1,L)$

$\delta(q_1,1) = (q_1,0,R)$

$\delta(q_1,B) = (q_f,B,R)$

elimination of null production :-

$$G = \begin{cases} S \to aS \mid AB \\ A \to \wedge \\ B \to \wedge \\ D \to b \end{cases}$$

$$G' = \begin{cases} S \to aS, \; S \to AB \\ S \to B, \; S \to A \\ S \to a \\ D \to b \end{cases}$$

Unit production :-

(Non terminal)

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

$C \to a$ (terminal)

⇓ Result

$G' = \boxed{S \to a}$

$A = BC$ (Non terminal & start)

$S = BC$ दो 2 variable

Topics :- i) Turing Machine recognisable Langauges

ii) Construction of PDA

(iii) Pumping lemma for CFL

(iv) Languages and their Automata

$x = y^a$, $y = z^b$ $z = x^c$

find abc

$x = y^{bc}$ $x = y^{1+1}$

$= z^{ab}$

$x = z^{abc}$ $y = z^1$

$z = x$

$a = b = c = 1$

x

$|uv| \leq |z|$

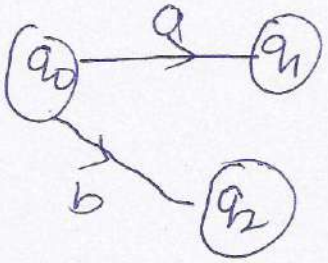
$uv^i w$

imp

Q diff^n b/w DFA and NFA

Path Unique in DFA

NFA



TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well