Date 2/10/18 Page No.:

Theory of Computation (7-10)

Course:-

50% Finite Automata and Regular languages
30% Push down Automata and Context free languages
20% { Linear bounded Automata and Context Sensitive languages
{ Turing Machines and Recursive Enumerable languages
{ Undecidability

Def: It is the mathematical study of computing machines and them capability

(or)

It is the study of Automata & formal languages.

Alphabet (Σ)

finite non-empty set of Symbols.

Ex:- $\Sigma = \{a, b, c\}$
$\Sigma = \{0, 1\}$
$\Sigma = \{\exists, \forall\}$

String : finite sequence of symbols over the given Σ.

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

a, ab ab, abbaab

"Education is the manifestation of perfection already in a man." –Swami Vivekananda

Scanned by CamScanner

Date __/__/____ Page No.: ____

Length of string

|ab| = 2 |a°b°| = 0

|abbab| = 5 ↓ ∈

∈ (epsilon) is a string of length 0.

Language :- Any set of strings over given Σ.

Ex: Σ = {0,1}

L_1 = {0, 1, 00, 01, 10, 11} → finite lang.

L_2 = {0, 1, 00, 01, ...} → infinite lang.

L_3 = { } → Empty language.

L_4 = {∈} → finite language with string of length 0.

L_5 = {∈, 0, 1, 00, 01, 10, 11, ...} = Σ^*

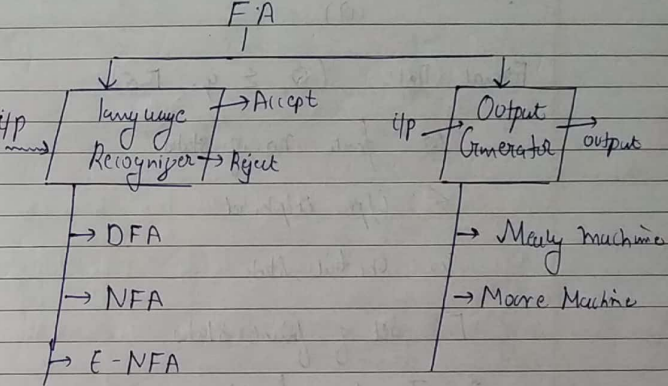
↑ Complete language

Date __/__/____ Page No.: ____

Finite Automata (F.A)

F.A is a mathematical model which contains finite no. of states and transitions defined b/w them.

There are 2 types of F.A known as



F.A can be represented by using

i) Transition Diagram
(or)
ii) Transition Table

Scanned by CamScanner

Page No.:

Date

Deterministic (F.A) (D.F.A)

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

It is a F.A in which from each and every state on every i/p symbol exactly 1 transition should exist.

(1)

Formal Def: $(Q, \Sigma, q_0, F, \delta)$

Q : finite no. of states

Σ : i/p alphabet

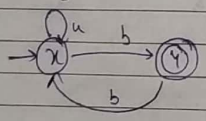
q_0 : initial state

F : Set of final states

δ : Transition function

$Q \times \Sigma \rightarrow Q$

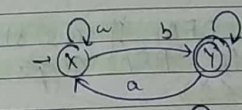
Q Identify valid FA?

1) 

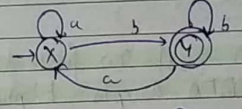
X

Date

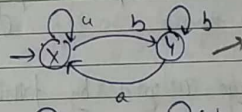
Page No.:

2) 

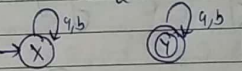
X

3) 

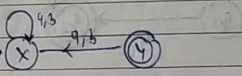
✓

4) 

→ accept Empty lang
✓

5) 

✓

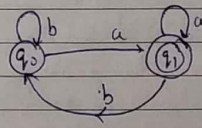
6) 

✓

Acceptance Method of D.F.A

By reading complete string if D.F.A results in final state then given string is accepted. (otherwise) string is rejected.

Set of all strings accepted by a D.F.A is known as Language of the D.F.A



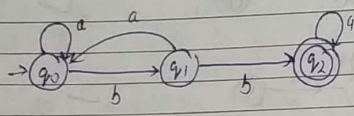
Scanned by CamScanner

Date __/__/____ Page No.:

NOTE 1) In any DFA if all states are final then the DFA accepts Complete lang (i.e. Σ^*)

2. In any DFA if no final state exists then the DFA accepts Empty lang.

Q Identify language accepted by following DFA.



Sol:

a) Set of all strings starting with bb

b) " " " " ending with bb

c) " " " " contains atleast 2 b's

d) None (ie containing bb as substring)

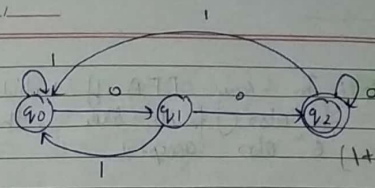
Sol: a) { bb, bbaa, bbbb, ... }

b) { abb, abbb, ... }

c) { bab, bbaa }

d) Substring "bb"

Date __/__/____ Page No.:

Q 

$(1+01+001)^* 000^*$

a) Set of all Starting with 0.

b) Set of all strings ending with 0.

c) " " " " containing Substring 00

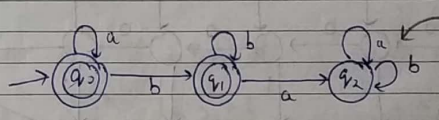
d) None (Set of all string ending with 00)

Sol: a) {0, 01} X X → Not accepted

b) {10, 010} X

c) {1001, 100100} X

d) —

Q 

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

b) $L = \{ a^n b^m / n \geq 0, m \geq 1 \}$

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

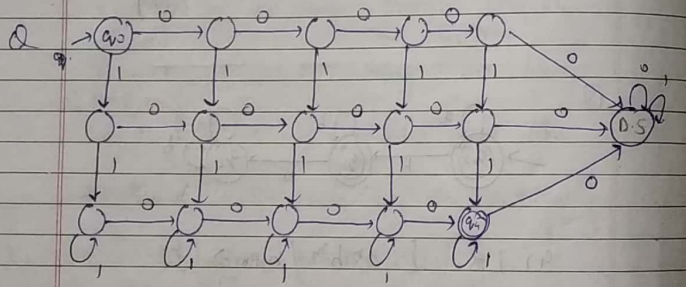
d) None

Scanned by CamScanner

Date __/__/____ Page No.: ____

NOTE: 1) In any DFA if initial state is also final state, then "ε" also accepted.

2) Dead state :- It is a non-final state in which self-loop exists on all i/p symbols



identify lang. accepted by the given DFA

a) Length of the string atleast 6.
b) No. of 0's at least 4. & no. of 1's at least 2.
c) No of 0's at least 4 & no. of 1's exactly 2
d) No of 0's exactly 4 & no of 1's at least 2.

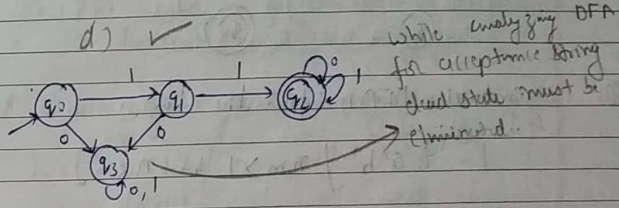
Date __/__/____ Page No.: ____

Sol: a) 0000000 ✗

b) 00110000 ✗

c) 00110000 ✗

d) ✓

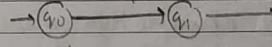


while analyzing DFA for acceptance string dead state must be eliminated.

How many n length strings this given automata accepts ?

		Substitute n	
a)	$\{2^n / n \geq 0\}$	$n=2$	2^2 ✗
b)	$\{2^{n-1} / n \geq 1\}$	$n=2$	2 ✗
c)	$\{2^{n-2} / n \geq 2\}$	$n=2$	1 ✓
d)	$\{2^{n-3} / n \geq 3\}$	$n \geq 3$	✗

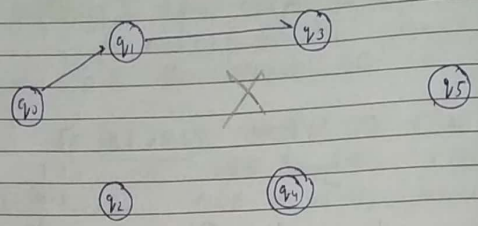
Sol: given DFA can be written like



Concept:- to reach final state, at least 2 1's we need to cross.

2 length string → $n=2$ only 1 string possible 11 — 1
3 length string → $n=3$ " 2 strings " 110, 111 — 2
$n=4$ " 3 " " 1100, 1101, 1110, 1110 — 4

Scanned by CamScanner



Q Construct DFA for the lang.

$L = \{ a^n b^m / n,m \geq 1 \text{ \&\& } n < m \}$

find out no. of states?

Sol:

DFA not possible bcz language is finite b/w symbols a & b

$a^n b^m$ $n < m$ i.e while comparing b we should make sure that no of b's is more than no of a's

F.A has finite memory, here we can not keep count of infinite no of b's

Drawback of FA

1) FA fails to accept languages in which comparison exists b/w symbols.

2) The languages for which F.A is possible are known as Regular Languages

Q Which of the following languages is Regular?

$L_1 = \{ a^n b^m / n,m \geq 1 \}$ R

$L_2 = \{ a^n b^n / n \geq 1 \}$ NR

$L_3 = \{ a^n b^n c^n / n \geq 1 \}$ NR

$L_4 = \{ a^n b^{2m} c^{3k} / n,m,k \geq 1 \}$ R

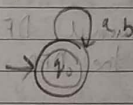
$L_5 = \{ a^n b^m c^{n+m} / n,m \geq 1 \}$ NR

$L_6 = \{ a^n b^m / n \neq m \}$ NR

$L_7 = \{ a^n b^m / n,m \geq 1 \text{ \& } n < m \}$ NR

Q Construct the DFA that accepts all strings of a's & b's, including ϵ.

Sol:



DFA ⟶ lang (1 to 1 mapping)

lang ⟶ DFA

Scanned by CamScanner

Date __/__/____ Page No.: ____

Minimization of DFA

1) For one regular lang many no of DFA's are possible but only minimal state DFA is only 1.

2) To construct minimal DFA 2 algorithms exists known as
   i) State Equivalence Algorithm
   ii) Marking Algorithm

3) These minimization algorithms are applicable only for DFA but not applicable for NFA & ε-NFA

4) Before applying minimization algorithm all the inaccessible states are eliminated from DFA

5) If any state is not reachable from initial state is known as inaccessible state ("it is different from dead state")

6) To construct minimal DFA equivalent states present in the DFA are merged into single state

"It is the prime responsibility of every citizen to love his country." —Sardar Vallabhai Patel

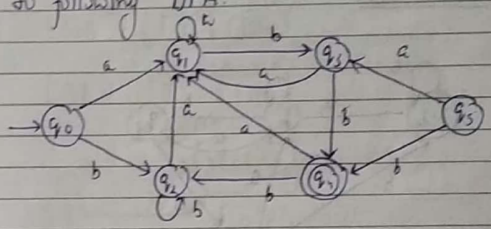
Date __/__/____ Page No.: ____

7) 2 State q_1 & q_2 are said to be equivalent if for all strings of complete lang

$\forall x \in \Sigma^*$ transition of q_1 on x could transition of q_2 on x results either Final (or) Non-final

$\delta(q_1, x)$ — F (or) NF — $\delta(q_2, x)$

Q) Construct the minimal state DFA, equivalent to following DFA.



Sol: Steps: 1) Check whether DFA is not.
2) eliminate inaccessible state
3) Make Transition table (T.T)

	a	b
q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
q_4	q_1	q_2

"Education is the manifestation of perfection already in a man." —Swami Vivekananda

Scanned by CamScanner

Date __/__/____ Page No.: ____

Check for equivalence states

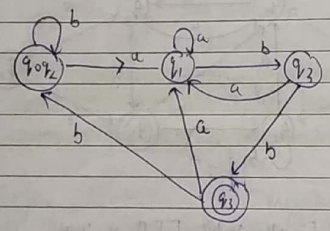
$\{q_0 q_1 q_2 q_3\}$ $\{q_4\}$

$\{q_0 q_1 q_2\}$ $\{q_3\}$ $\{q_4\}$

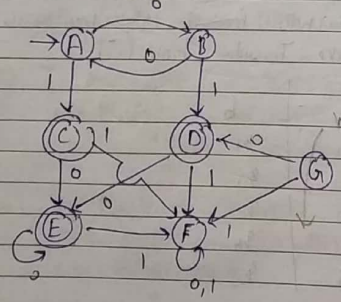
$\{q_0 q_2\}$ $\{q_1\}$ $\{q_3\}$ $\{q_4\}$

$\{q_0 q_2\}$ $\{q_1\}$ $\{q_3\}$ $\{q_4\}$

Minimize DFA



Q



unreachable

Date __/__/____ Page No.: ____

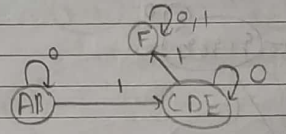
T.T

	0	1
→ A	B	C
B	A	D
*C	E	F
*D	E	F
*E	E	F
F	F	F

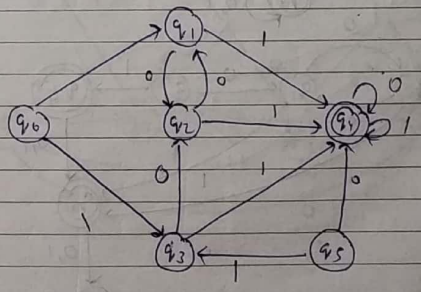
$\{ABF\}$ $\{CDE\}$

$\{AB\}$ $\{F\}$ $\{CDE\}$

$\{AB\}$ $\{F\}$ $\{CDE\}$



Q



Scanned by CamScanner

Date __/__/____ Page No.: ____

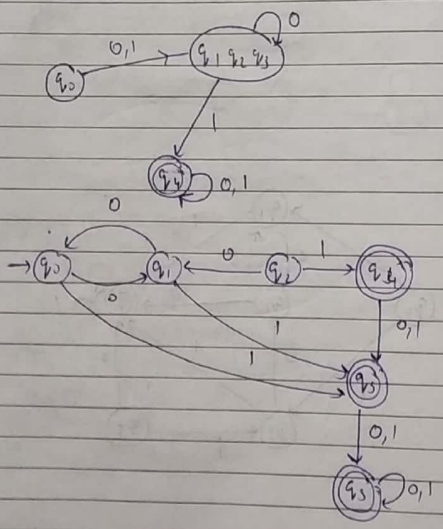
T.T

	0	1
→ q_0	q_1	q_3
q_1	q_2	q_4
q_2	q_1	q_4
q_3	q_2	q_4
* q_4	q_4	q_4

$\{q_0 q_1 q_2 q_3\}$ $\{q_4\}$

$\{q_0\}$ $\{q_1 q_2 q_3\}$ $\{q_4\}$

$\{q_0\}$ $\{q_1 q_2 q_3\}$ $\{q_4\}$



Date __/__/____ Page No.: ____

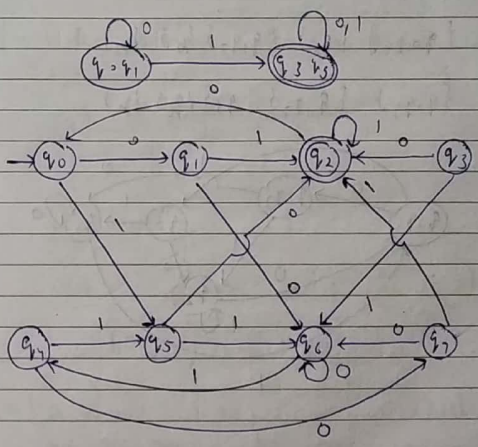
T.T

	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3
q_3	q_5	q_5
* q_5	q_5	q_5

$\{q_0 q_1 q_3\}$ $\{q_5\}$

$\{q_0 q_1\}$ $\{q_3 q_5\}$

$\{q_0 q_1\}$ $\{q_3 q_5\}$



Scanned by CamScanner

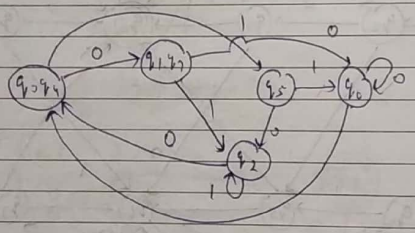
	0	1
q_0	q_1	q_5
q_1	q_6	q_2
* q_2	q_0	q_2
q_3	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

$\{q_0 q_1 q_3 q_5 q_6 q_7\}$ $\{q_2\}$

$\{q_0 q_3 q_6\}$ $\{q_1 q_7\}$ $\{q_5\}$ $\{q_2\}$

$\{q_0 q_3\}$ $\{q_6\}$ $\{q_1 q_7\}$ $\{q_5\}$ $\{q_2\}$

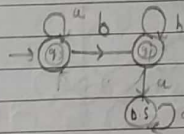
$\{q_0 q_3\}$ $\{q_1 q_7\}$ $\{q_5\}$ $\{q_6\}$ $\{q_2\}$



Q Construct minimal state DFA for the following language

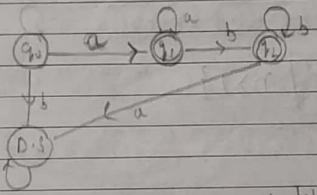
$L = \{ a^n b^m / n, m \geq 0 \}$ (any no of a's followed by any no of b's.)

Sol. $L = \{ \epsilon, a, b, aa, ..., bb, bbb, ..., ab, aab, abb, ... \}$



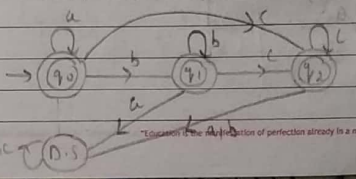
Q $L = \{ a^n b^m / n \geq 1, m \geq 0 \}$

Sol. $L = \{ a, aa, aaa, ... ab, abb, abbb, ... \}$



Q $L = \{ a^n b^m c^k / n, m, k \geq 0 \}$ (any no of a's followed by any no of b's followed by any no of c's)

Sol. $L = \{ \epsilon, a, aa, aaa, ... b, bb, ..., c, cc, ... ab, aabb, ... ac, acc, aacc, ... bc, bbcc, ... \}$



Scanned by CamScanner

Q $L_4 = \{ a^n b^m / n \geq 2, m \geq 0 \}$

Sol

Q $L_5 = \{ a^{(n)} / n \geq 1 \}$

Sol

*Q $L_6 = \{ a^{n^2} / n \geq 1 \}$

Sol: $L = \{ a^1, a^2, a^3, \ldots \}$

DFA not possible

Q $L_7 = \{ a^{2^n} / n \geq 1 \}$

Sol DFA not possible

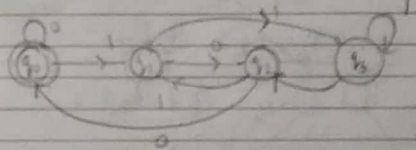
$L = \{ a^1, a^2, a^3, \ldots \}$

NOTE: FA fails to accept languages in which strings formed by the symbols are not having common difference.

Q Construct a minimal state DFA that accepts set of all binary nos. which are divisible by 4.

Sol: $\Sigma = \{0, 1\}$ $L = \{0100, 1000, 1100, \ldots\}$

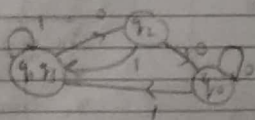
	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_2	q_3



$\{q_1, q_2, q_3\}$ $\{q_0\}$

$\{q_0, q_1\}$ $\{q_2\}$ $\{q_3\}$

$\{q_0, q_3\}$ $\{q_1\}$ $\{q_2\}$

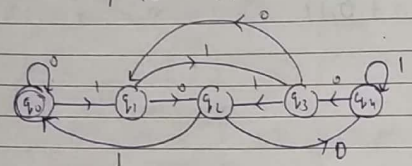


Scanned by CamScanner

Q Binary no's divisible by 5

Sol.

q	0	1
→ q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4



Q Binary no's divisible by 6

Sol.

	0	1
→ q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_5
q_3	q_0	q_1
q_4	q_2	q_3
q_5	q_4	q_5

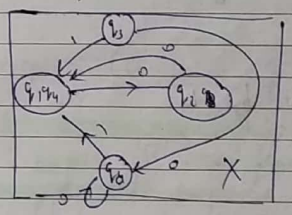
Minimizing

$\{q_1 q_2 q_3 q_4 q_5\}$ $\{q_0\}$

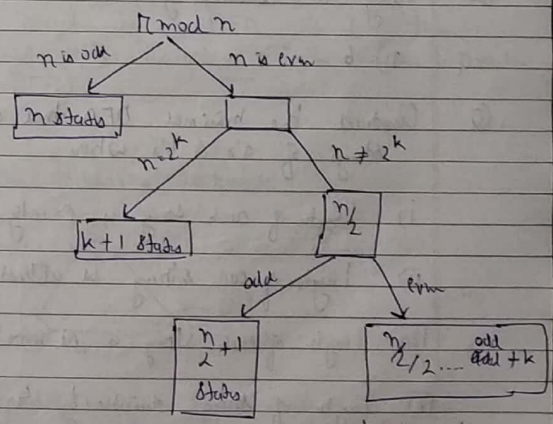
$\{q_1 q_2 q_4 q_5\}$ $\{q_3\}$ $\{q_0\}$

$\{q_1 q_4\}$ $\{q_2 q_3 q_5\}$ $\{q_0\}$

$\{q_1 q_4\}$ $\{q_2 q_3\}$ $\{q_5\}$ $\{q_0\}$



**Note:-



k = no of times division done

Scanned by CamScanner

Q Binary no's divisible by 8

Sol: $n = 4$

Q Binary no's divisible by 9

$n = 9$

Q Binary no's divisible by 12

$n = 5$

Q Binary no's divisible by 16

$n = 5$

Q Binary no's divisible by 24

$n = 6$

Q Construct the minimal DFA that accepts all strings of a's & b's where

i) Length of each string is exactly 4

ii) Length of each string is atleast 4

iii) Length of each string is at most 4.

iv) Length of string divisible by 4

v) No's of a's divisible by 3.

vi) a's divisible by 2 and b's divisible by 3

vii) a's are even and b's are odd

viii) a's divisible by 2 or b's divisible by 3

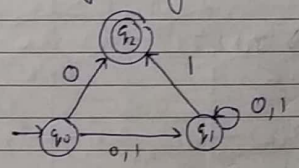
ix) a's odd (or) b's are even

Concept:- Complement of DFA

→ by interchanging final state as non-final and non-final state as final we can get DFA complement.

→ If lang. accepted by a DFA is L, then complemented DFA accepts $\Sigma^* - L$ (this is possible only for DFA)

Q Consider the following automata



Lang accepted by given automata is L. Then interchange final & non-final states of given automata, then language accepted by resultant automata is L_1

"It is the prime responsibility of every citizen to feel that his country" –Sardar Vallabhai Patel

"Education is the manifestation of perfection already in a man." –Swami Vivekananda

Scanned by CamScanner

Date __/__/____ Page No.:

Which of the following is True?

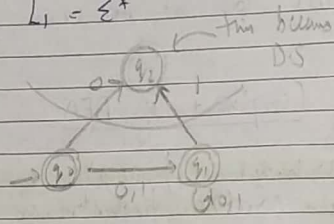
a) $L_1 = L$

b) $L_1 = \Sigma^* - L$

c) $L_1 \subseteq L$

d) $L_1 = \Sigma^*$

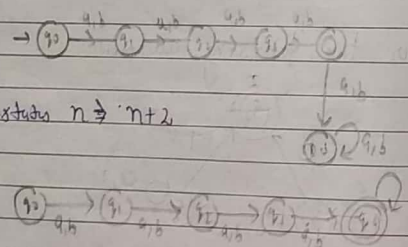
Sol:



now it accepts ε also $L_1 = \Sigma^*$

Q2

Sol: i)



no of states $n \rightarrow n+2$

ii)

no of states = n+1

"It is the prime responsibility of every citizen to feel that his country." —Sardar Vallabhai Patel

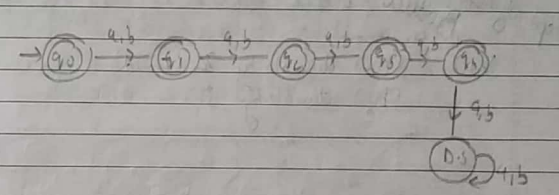
Date __/__/____ Page No.:

NOTE: The minimal DFA that accepts set of all strings, where length of each string is exactly n. requires n+2 states.

* # length of string not exactly n will also have n+2 states

* Minimal DFA that accepts set of all strings, where length of string is at least n requires n+1 states

iii) length at most 4

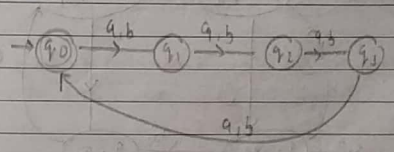


NOTE:

# of states = n+2

in minimal DFA that accepts all strings where length of each string is at most n

iv) length divisible by 4



# of states in minimal DFA = n

"Education is the manifestation of perfection already in a man." —Swami Vivekananda

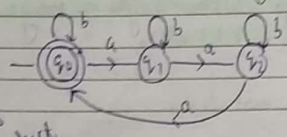
Scanned by CamScanner

NOTE

→ The minimal DFA that accept strings of length of string divisible by k will contain n states.

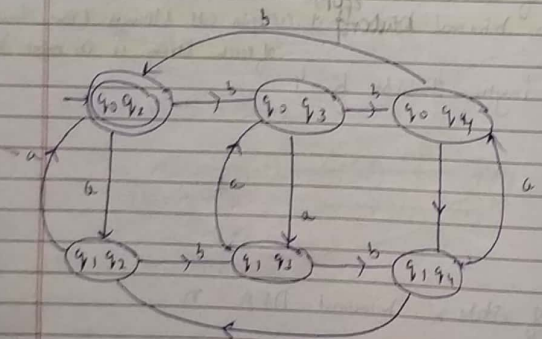
→ Not divisible by k will also contain n states

v) n_a divisible by 3



(concept of Cross Product

vi) (n_a) divisible by 2 (consult)
(n_b) divisible by 3

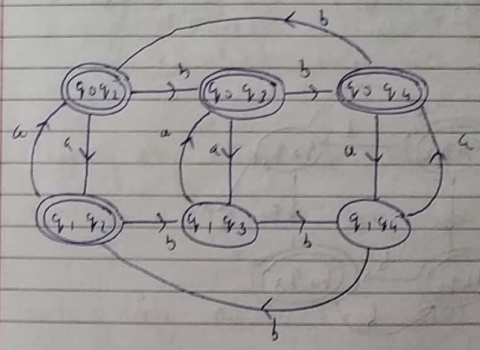


initial state = initial state of both the automata

final state = final state of both the automata

viii) (n_a) divisible by 2 (OR) take (row product
(n_b) divisible by 3

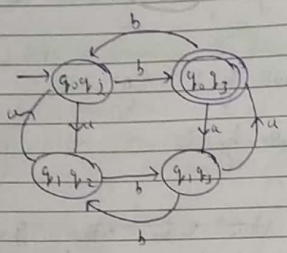
Here Final state means either q_0 is present (or) q_2 is present i.e if final state of either of automata is present then make the state as final state



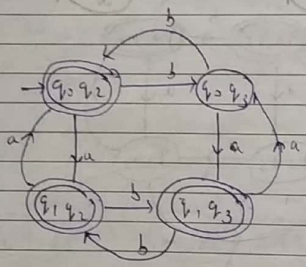
vii) If (n_a) is even
(consult)
(n_b) is odd

Scanned by CamScanner

Date / Page No.:



(i) (n_a) is odd (OR) (n_b) are even



Q (n_a) divisible by 2 (AND) (n_b) not divisible by 3

Sol: $n = 6 \quad \leftarrow 2 \rightarrow 3$

"It is the prime responsibility of every citizen to feel that his country." –Sardar Vallabhai Patel

Date / Page No.:

[Concept: AND = LCM(m,n) in case of divisibility
OR = GCD(m,n)]

Q (n_a) exactly 2 (AND) (n_b) at least 3

Sol: take 3 (remove D.S) 4

add D.S after multiplication

$n \Rightarrow 3 \times 4 = 12 + 1 \rightarrow 13$

Q a's exactly 4 (AND) b's at least 2

Sol: $5 \times 3 \Rightarrow 15 + 1 = 16$

Q a's exactly 3 (OR) b's at most 2

Sol: $4 \times 3 = 12 + 1 = 13$

Q a's divisible by 6 & b's not divisible by 8

Sol: $6 \times 8 = 48$

Q length of string divisible by 2 (OR) divisible by 4

Sol: $n = 4$ ie LCM(2,4) bez GCD(2,4) $\neq 1$

Q length of string divisible by 2 (and) divisible by 4

$n = 4 \quad$ LCM(2,4)

Q length of string divisible by 6 (and) divisible by 8

$n = 24 \quad$ LCM(6,8)

"Education is the manifestation of perfection already in a man." –Swami Vivekananda

Scanned by CamScanner

Q Length of string divisible by 4 and 6

$n = 12$ $\quad LCM(4,6)$

Q Length of string divisible by 3 & 5 Ans: 15

(Concept): $GCD(m,n) = 1$ AND / OR $m \times n$

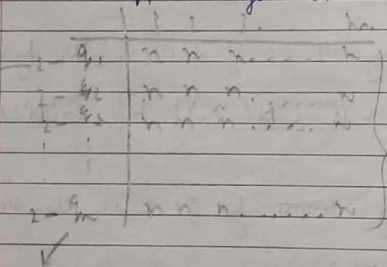
CQ Length of string divisible by 4 & 6 Ans: 12

(Concept): $GCD(m,n) \neq 1$ AND / OR $LCM(m,n)$

**

Q How many maximum no. of DFA's can be constructed with n states $\{q_1, q_2, q_3, \ldots q_n\}$ over i/p alphabet $\Sigma = \{1, 2, 3, \ldots m\}$ contain m symbols. Where q_1 is always initial state.

Sol:-



$2^n \times (n)^{nm}$

# of DFA = $2^n \times (n)^{nm}$

"It is the prime responsibility of every citizen to feel that his country." –Sardar Vallabhai Patel

Q How many maximum no. of DFA can be constructed with 2 states X & Y over the i/p alphabet $\Sigma = \{a, b\}$ that accepts complete language?

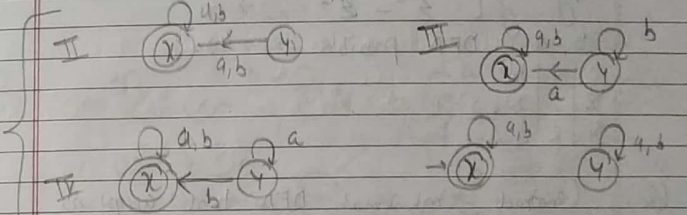
Here fix X as initial state.

Sol. 1-case

	a	b
→ X	2	2
Y	2	2

$2^4 = 16$

II, III, IV



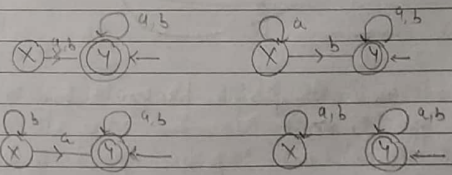
$16 + 4 = 20 + 20 = 40$

II case

	a	b
X	2	2
→ Y	2	2

$2^4 = 16$

Here fix Y as initial state.



"Education is the manifestation of perfection already in a man." –Swami Vivekananda

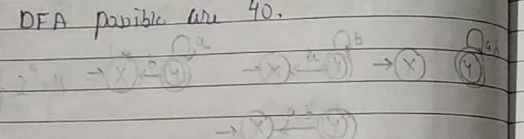
Scanned by CamScanner

Q How many maximum no. of DFA can be constructed with 2 states X & Y over i/p alphabet $\Sigma = \{a, b\}$ that accepts empty language.

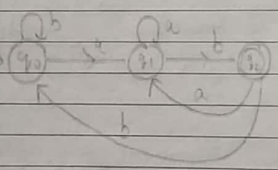
The DFA need to accept the empty lang. if we take complement of DFA accepting complete lang then we get empty lang.

If $L = \Sigma^* - \Sigma^* = \{\}$

∴ # of DFA possible are 40.



Q Construct minimal DFA that accept all strings of a's and b's where each string ending with ab.



L = {ab, abab, babab, bbabab, ...}

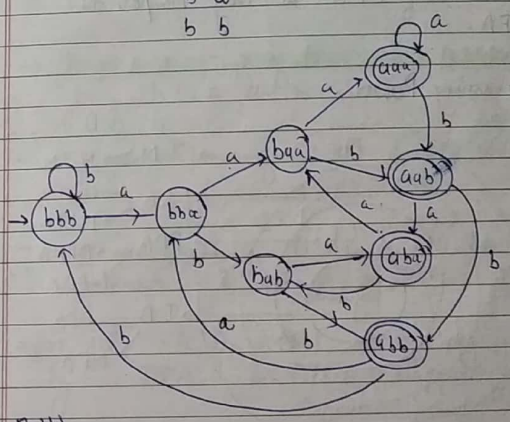
**

Q Construct the minimal DFA that accepts all strings of a's and b's where 3rd i/p symbol is "a" while reading the string from right hand side.

Ending possibility

a _ _

a a
a b
b a
b b



i/p
a
aaaa
aaab b

i/p
a
abba
aabb b

i/p
abba a
abbb b

R.H.S

3 → 8
4 → 16
5 → 32
⋮
8 → 256
⋮
n → 2^n

# of states in minimal DFA where n^{th} i/p symbol is "a" while reading string from right hand side is 2^n

Scanned by CamScanner

Date / /

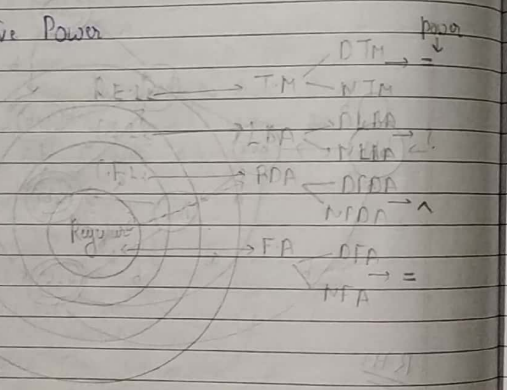
Page No.

NOTE: 1) The minimal DFA that accepts all strings of a's and b's where nth symbol is 'a' while reading the string from right hand side requires 2^n states.

2) For some regular languages construction of DFA is difficult, hence NFA can be constructed.

3) Construction of NFA is easy compare to DFA.

Expressive Power



Def:- no. of languages accepted by a particular machine is known as expressive power of that machine

Date 3/10/18

Page No.

TM > LBA > PDA > FA

NOTE:

Expressive power of TM is more than LBA, PDA and FA

Expressive power of PDA is more than FA.

→ Expressive power of DFA & NFA is same hence every NFA is converted into DFA

→ Expressive power of NPDA > DPDA. Hence every NPDA can not be converted into DPDA.

→ Expressive power of DTM & ~~NTM~~ NTM is same.

→ Expressive power of DLBA & NDLBA is unknown (?)

Q Let D_f & D_p are no. of language accepted by DFA & DPDA respectively.

Let N_f & N_p are no. of languages accepted by NFA & NPDA respectively.

Which of the following is True?

Scanned by CamScanner

a) $D_f \subseteq N_f$ & $D_p = N_p$

b) $D_f \subset N_f$ & $D_p \in N_p$

c) $D_f = N_f$ & $D_p \supset N_p$

d) None.

Non-Deterministic F.A

↳ In NFA from the given state, given i/p symbol there may be zero no. of transitions (or) one (or) more transitions.

↳ Formal definition of NFA is :-

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

Q: finite no. of states
Σ: i/p alphabet
q_0: initial state
F: ~~finite state~~ set of final states
δ: Transition fun

$$Q \times \Sigma \rightarrow 2^Q$$

Note:

Every DFA is NFA but every NFA need not be DFA.

Minimization algorithms are not applicable for NFA.

If language accepted by NFA is L, then by interchanging final and non-final states resultant NFA may (or) may not accept $\Sigma^* - L$

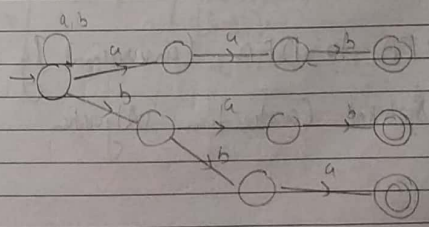
For valid strings also NFA may halts in non-final states.

Language recognization is easy in DFA, compared to NFA.

Construction of NFA is easy compared to DFA.

Q Construct NFA that accepts all strings of a's & b's where each string is ending with "aab" (or) "bab" (or) "bba" ?

Sol:



Scanned by CamScanner

Date __/__/____ Page No.:

Q Each string starting and ending with different symbol

Sol

Q Each string contains "aaa" (or) "bbb" as substring

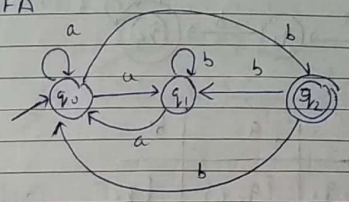
Sol

Q 8th i/p symbol is "a" while reading the string from R.H.S

NFA to DFA Conversion by using Subset Construction Algorithm

Date __/__/____ Page No.:

Q Construct an equivalent DFA for the following NFA



Sol

NFA

T.T	a	b
q_0	$\{q_0, q_1\}$	q_2
q_1	q_0	q_1
q_2	—	$\{q_0, q_1\}$

DFA T.T

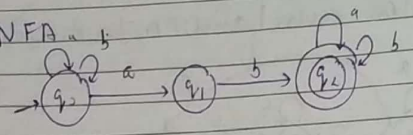
	a	b
→ q_0	$[q_0 q_1]$	q_2
$[q_0 q_1]$	$[q_0 q_1]$	$[q_1 q_2]$
q_2	q_D	$[q_0 q_1]$
$[q_1 q_2]$	q_0	$[q_0 q_1]$
q_D	q_D	q_D

q_D = dead state

Scanned by CamScanner

Date __/__/____ Page No.:

NFA =



Sol:

T.T	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_0$
q_1	—	q_2
q_2	q_2	q_2

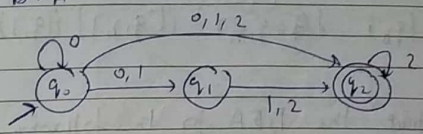
D.F.A T.T

	a	b
→ q_0	$[q_0 q_1]$	q_0
$[q_0 q_1]$	$[q_0 q_1]$	$[q_0 q_2]$
$[q_0 q_2]$	$[q_0 q_1 q_2]$	$[q_0 q_2]$
$[q_0 q_1 q_2]$	$[q_0 q_1 q_2]$	$[q_0 q_2]$

"It is the prime responsibility of every citizen to feel that his country." —Sardar Vallabhai Patel

Date __/__/____ Page No.:

Q Construct minimal DFA for the following NFA



Sol. NFA T.T

	0	1	2
→ q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	q_2
q_1	—	q_2	q_2
q_2	—	—	q_2

T.T for DFA

	0	1	2
→ q_0	$[q_0 q_1 q_2]$	$[q_1 q_2]$	q_2
$[q_0 q_1 q_2]$	$[q_0 q_1 q_2]$	$[q_1 q_2]$	q_2
$[q_1 q_2]$	q_D	$[q_1 q_2]$	q_2
q_2	q_D	q_D	q_2
q_D	q_D	q_D	q_D

"Education is the manifestation of perfection already in a man." —Swami Vivekananda

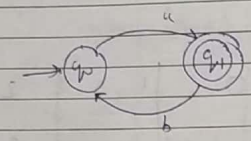
Scanned by CamScanner

Date __/__/____ Page No.: ____

$\{q_0, q_D\}$ $\{[q_0 q_1 q_2]\ [q_1 q_2]\ q_2\}$

$\{q_0\}\ \{q_D\}\ \{q_0 q_1 q_2\}\ \{[q_1 q_2]\}\ \{[q_1]\}$

Q Construct the DFA for the following NFA



Sol. T.T

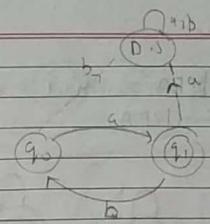
	a	b
→ q_0	q_1	—
q_1	—	q_0

DFA T.T

	a	b
→ q_0	q_1	q_D
q_1	q_D	q_0
q_D	q_D	q_D

"It is the prime responsibility of every citizen to feel that his country." –Sardar Vallabhai Patel

Date __/__/____ Page No.: ____



NOTE: While converting NFA into DFA for the given "n" states NFA no. of states possible in DFA is 1 to 2^n. min → 1 max → 2^n

ϵ - NFA

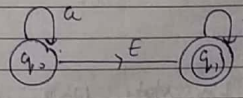
NFA having ϵ transitions is known as ϵ - NFA.

(or)

Formal Definition:

ϵ - NFA = $(Q, \Sigma, q_0, F, \delta)$

$\delta: Q \times \Sigma \cup \{\epsilon\} \longrightarrow 2^Q$

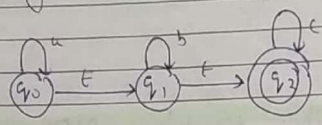


"Education is the manifestation of perfection already in a man." –Swami Vivekananda

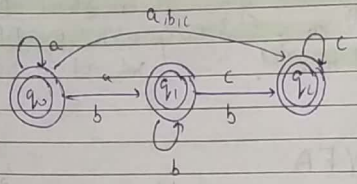
Scanned by CamScanner

Date __/__/____ Page No.: ____

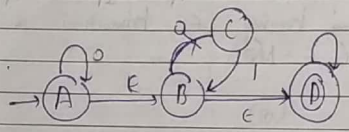
Q Construct NFA without "ε" for the following ε-NFA

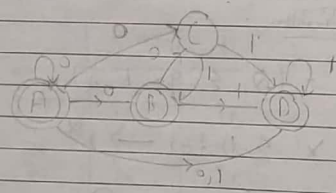


Sol:



Q

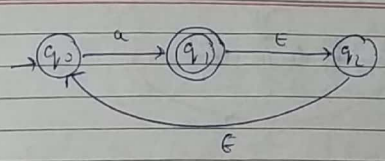




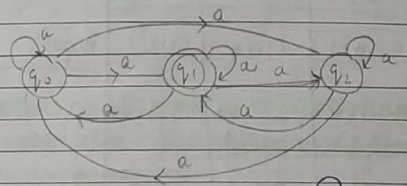
NOTE: Those states where which can be reached just by using ε moves, are considered to be final state. (~~from initial~~ State)

Date __/__/____ Page No.: ____

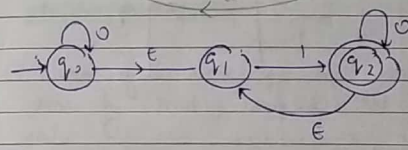
(2)

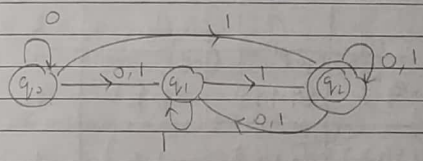


$q_1 \rightarrow \varepsilon\,\varepsilon\,a\,\varepsilon \rightarrow q_2$

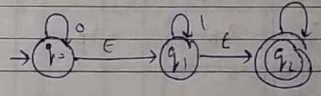


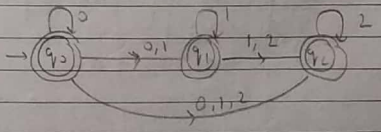
Q



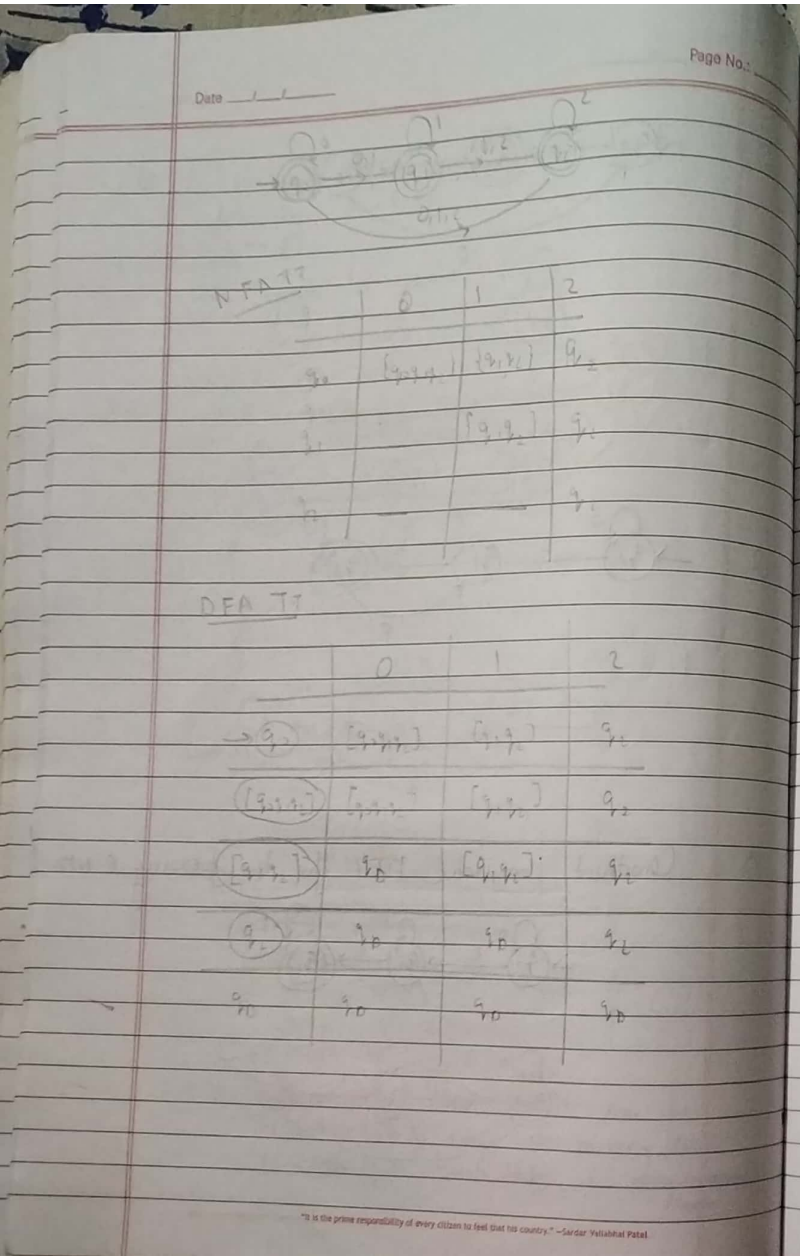


Q Construct minimal DFA for the following ε-NFA





Scanned by CamScanner



NFA TT

	0	1	2
q_0	$[q_0,q_1]$	$[q_1,q_2]$	q_2
q_1		$[q_1,q_2]$	q_2
q_2			q_2

DFA TT

	0	1	2
$\rightarrow q_0$	$[q_0,q_1]$	$[q_1,q_2]$	q_2
$[q_0,q_1]$	$[q_0,q_1]$	$[q_1,q_2]$	q_2
$[q_1,q_2]$	q_D	$[q_1,q_2]$	q_2
q_2	q_D	q_D	q_2
q_D	q_D	q_D	q_D

q_D {q_0 $[q_0,q_1]$ $[q_1,q_2]$ q_2}

q_D {q_0 $[q_0,q_1,q_2]$ $[q_1,q_2]$ q_2}

ϵ-closure

While converting from ϵ-NFA to DFA initial state of DFA is ϵ-closure of NFA initial state.

ϵ-closure of a state q is starting from that state by reading only ϵ set of all reachable states.

Ex: ϵ-closure $(q_0) \rightarrow \{q_0, q_1, q_2\}$

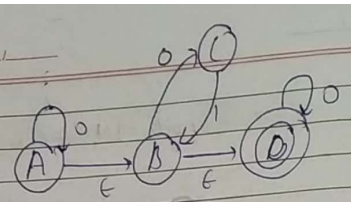
ϵ-closure $(q_1) \rightarrow \{q_1, q_2\}$

ϵ-closure $(q_2) \rightarrow \{q_2\}$

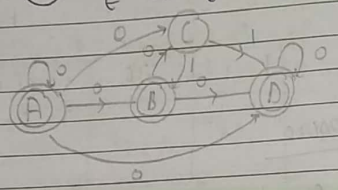
	0	1	2
$[q_0,q_1,q_2]$	$[q_0,q_1,q_2]$	$[q_1,q_2]$	q_2
$[q_1,q_2]$	q_D	$[q_1,q_2]$	q_2
q_2	q_D	q_D	q_2
q_D	q_D	q_D	q_D

Scanned by CamScanner

Q



Sol.



	0	1
A	{A, B, C, D}	
B	{C, D}	
C		{B, D}
D	D	

ε-closure (A) → ABD

	0	1
[ABD]	[ABCD]	q_D
[ABCD]	[ABCD]	[BD]
q_D	q_D	q_D
[BD]	[CD]	q_D
[CD]	D	[BD]
D	D	q_D

q_D {[ABD] [ABCD] [BD] [CD] D}

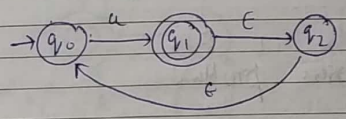
q_D {[ABD] [CD] [BD] D} [ABD]

q_D {[ABD] [CD] D} {[BD]} {ABCD}

{q_D} {[ABD], D} [CD] [BD] {ABCD}

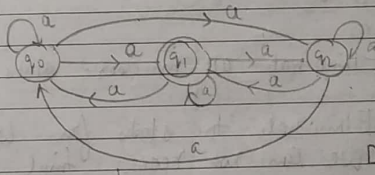
Q What is the complement of the language accepted by following ε-NFA?

Sol.



Ans: ε

Sol. Concept:- Convert to DFA



T.T

	a
q_0	{q_0, q_1, q_2}
q_1	{q_1, q_2}
q_2	{q_2}

DFA T.T

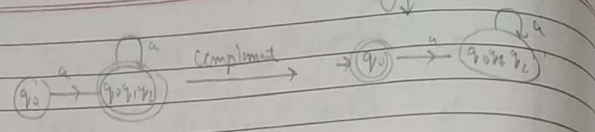
	q
q_0	[$q_0q_1q_2$]
[$q_0q_1q_2$]	[q_1q_2]

ε-closure (q_0) → q_0

Scanned by CamScanner

Date __/__/____ Page No.:

this DFA accepts only ε



Complement of lang is ε

Decision Properties of F.A (Imp)

1) finiteness problem

means checking whether lang. accepted by given automata is finite or not

To check

i) Eliminate all inaccessible states

ii) Eliminate the states from which we can not reach final

iii) In the resultant automata if there exists loop (or) cycle then automata accepts infinite lang. otherwise finite lang

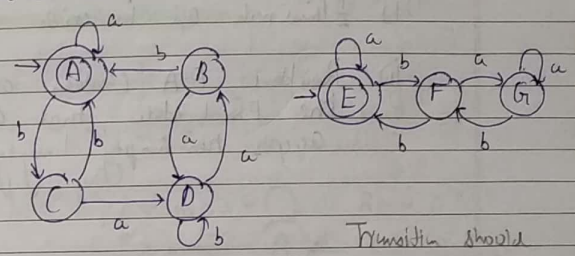
"It is the prime responsibility of every citizen to feel that his country." —Sardar Vallabhai Patel

Date __/__/____ Page No.:

2) Equivalence Problem

means checking whether given two finite automata accepts same lang. (or) not.

Q) check whether following 2 automata are equal (or) not..



	a	b
(AE)	(AE)	(CF)
(CF)	(DG)	(AE)
(DG)	(BG)	(DF)
(BG)	(DG)	(AE)

Transition should result either (F,F) (or) (NF, NF)

F → final
NF → non-final

if (F, NF) (or) (NF, F) then both FA's are not equal

↑ final ↑ non-final

Hence they are not equivalent

Concept - take initial state of both the F.A and find the transitions for i/p symbols

"Education is the manifestation of perfection already in a man." —Swami Vivekananda

Scanned by CamScanner

Date __/__/__ Page No.:

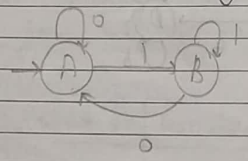
Emptiness Problem

means checking whether lang accepted by given FA is Empty language or not.

To check

i) Eliminate all inaccessible states

ii) Resultant FA contains at least one Final state, that automata accepts non-empty language.



Regular Expression

i) Simplest way of representing Regular languages is known as Regular expression

ii) For Every R.L, regular expression is constructed named every Regular expression represents one R.L. Hence there is no difference b/w R.L & Regular expression

Date 4, 10, 18 Page No.:

For Every R.L, regular expression can be constructed by using i/p symbols and following 3 operators.

+ => Union operator
. => Concatenation operator
* => Kleen closure operator

For ~~non-R.L~~ one R.L, many regular expressions can be possible.

Q Construct R.E for the following languages

i) $L_1 = \{a^n b^n / n \geq 0\}$ not possible.

$L_2 = \{a^n b^m / n, m \geq 0\}$ $RE \rightarrow a^* b^*$

$L_3 = \{a^n b^m / n \geq 1, m \geq 0\}$ $R.E \rightarrow a^+ b^*$

$a^+ = a \cdot a^*$

$L_4 = \{a^n b^m / n \geq 3, m \geq 2\}$ $R.E \rightarrow aaa^+ bb^+$

$aaaa^* bbb^*$

$L_5 = \{a^n b^m / n < m\}$ not possible

* $L_6 = \{a^n b^m / n > m \text{ and } n < m\}$ $RE = \{\} \Rightarrow \phi$

* $L_7 = \{a^n b^m / n > m \text{ (or) } n < m\} = \{aab, abb, \ldots\}$ not possible $= \{a^n b^m / n \neq m\}$

$L_8 = \{a^n b^m / n \geq m \text{ and } n < m\} = \{a^n b^n / n \geq 0\}$ not possible

Scanned by CamScanner

Date / /

Page No.:

$L_9 = \{ \in, a, b, aa, ab, ba, bb, \ldots \}$

$RE = (a+b)^*$

$L_{10} = \{ a, aa, aaa, \ldots \}$

$RE = a^+$

$L_{11} = \{ a^n b^m / (n+m) \text{ is even} \}$

$RE = (aa)^* (bb)^* + a(aa)^* b(bb)^*$

$L_{12} = \{ a^n b^m / (n+m) \text{ is odd} \}$

$RE = (aa)^* b(bb)^* + a(aa)^* (bb)^*$

$L_{13} = \{ 1, 2, 4, 8, \ldots 2^n, \ldots \}$ all the no's are in binary

Let symbol be 1

$\{ 1, 11, 1111, 11111111, \ldots \} = 1^{2^n}$

(i/p)

Common difference $\neq$ constant

Hence RE not possible

$L_{14} = n_a(w) \bmod 3 \leq n_b(w) \bmod 3 \rightarrow$ Regular language

L_{14} is R.L bcz in case of modulo division count can be done in finite memory. (with)

"It is the prime responsibility of every citizen to feel that his country." --Sardar Vallabhai Patel

Date / /

Page No.:

$L_{14} = \{ 1, 2, 4, 8, 2 \ldots 2^n \ldots \}$ all the numbers are in binary

$\{ 1, 10, 100, 1000, \ldots \}$

10^* (or) 0^*10^*

Q Find the R.E that generates set of all odd length palindrome strings over i/p alphabet $\Sigma = \{a\}$

Sol $L = \{ a, aaa, aaaaa, \ldots \}$

$RE = a(aa)^*$

Q Find the R.E that generates set of all even length palindrome strings over i/p alphabet $\Sigma = \{a, b\}$

Sol Not possible bcz given language is not regular language

Q Find the R.E that generates set of all odd length palindrome strings of English language

S. Not possible

"Education is the manifestation of perfection already in a man." --Swami Vivekananda

Scanned by CamScanner

Date __/__/____ Page No.: ____

Note: palindrome languages formed over more than one symbol are not regular hence R.E is not possible

Q Construct the R.E that generates set of all even length palindromes strings over of Tamil lang.

Sol. Not possible.

Q Construct R.E for the following languages

$L_1 = \{ ww^R / w \in \{a,b\}^* \}$

$\{\epsilon, aa, aaaa, \ldots\}$

$= (aa)^*$

$L_2 = \{ ww^R / w \in \{a,b\}^* \}$ → set of palindrome strings over 2 i/p symbols

not possible bcz general strings are comparison based strings

$L_3 = \{ wxw^R / w \in \{a,b\}^* \}$ put $w = \epsilon$

$\epsilon x \epsilon = x$ → does not belongs to L. Hence not regular.

not possible.

$L_5 = \{ wxw^R / w, x \in \{a,b\}^+ \}$

let $w = \epsilon$ $x = (a+b)^*$ is complete lang.

$w^R = \epsilon$

"It is the prime responsibility of every citizen to feel that his country." —Sardar Vallabhai Patel

Date __/__/____ Page No.: ____

$w = ab$ $w^R = ba$

$ab(a+b)^* ba \in$ Complete lang.

Hence $\{wxw^R / w, x \in \{a,b\}\} \rightarrow R.E = (a+b)^*$

$L_5 = \{ ww / w \in \{a,b\}^* \}$

$w w \rightarrow$ comparison exists

$ab\ ab$ → not possible

$L_6 = \{ ww / w \in \{a,b\}^* \}$

$\{\epsilon, aa, aaaa, \ldots\} \Rightarrow (aa)^*$

$L_7 = \{ ww^R x / w, x \in \{a,b\}^* \}$

$x \in \epsilon$

$\{\epsilon, a, b, aa, bb, abba, baba, \ldots\} = (a+b)^*$

$L_8 = \{ wxw^R / w, x \in \{a,b\}^+ \}$

$a(a+b)^+ a + b(a+b)^+ b.$

Properties (Important)

1) $R + \phi = \phi + R = R$

2) $R \cdot \epsilon = \epsilon \cdot R = R$

3) $R \cdot \phi = \phi \cdot R = \phi$

$\{\epsilon, R\} \neq \{R\}$

4) $R + \epsilon \neq R$

"Education is the manifestation of perfection already in a man." —Swami Vivekananda

Scanned by CamScanner

Date __/__/____ Page No.:

5) $a+b = b+a$

6) $a \cdot b \neq ba$

7) $R^* = (R^*)^* = (R^*)^+ = (R^+)^*$

8) $RR^* = R^+ = R^*R$

9) $RR^* + \epsilon = R^+ + \epsilon = R^*$

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

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

11) $(a+b)^* \neq a^* + b^*$

12) $(a+b)^* \neq (ab)^*$

13) $(a+b)^* \neq a^*b^*$

14) $a^* + a = a^*$

15) $a(ba)^* = (ab)^*a$

16) $\epsilon^* = \epsilon$
$\epsilon^+ = \epsilon$

17) $\phi^* = \epsilon$
$\phi^+ = \phi$

Q What is the lang generated by following RE

$b^*(a^* \cdot \phi \cdot b + ab + a \cdot \phi^* \cdot b^*)(b+\phi)^*$

$b^*(\phi + ab \cdot + ab^*)b^*$

$b^*(ab + ab^*)b^*$

$b^* a b^*$

"It is the prime responsibility of every citizen to feel that his country." –Sardar Vallabhai Patel

Date __/__/____ Page No.:

a) exactly one a b) at ~~least~~ least one a

c) atmost one a d) None

Q $L_1 = a \quad L_2 = \phi$.

$L_1^* \cdot L_2 + L_2^*$

$a^* \cdot \phi + \phi^* \Rightarrow \phi + \phi^* \Rightarrow \phi^* = \epsilon$

Note :- Concatination operator having higher precedence than union operator

2) Kleen closure operator is having higher precedence than Concatination operator

$* > \cdot > +$

Q48 P-9

Sol: $\lambda + 1^*(011)^*(1^*(011)^*)^*$

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

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

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

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

Concept

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

ie $(a+b)^*$ covers a^*b^* also

"Education is the manifestation of perfection already in a man." –Swami Vivekananda

Scanned by CamScanner

Q19 P-8

Sol. $a^*b^*(a^*b^*)^*$

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

Q20 P-6

Sol. $R_1 = 11(0+1)^*$ → {11, 011, 111, ...}

$R_2 = (0+1)^*11$

$R_3 = 11(0+1)^*11 + 111 + 11$

$R_1 \supset R_3$ bcz R_1 can generate 110, 1110, ... which R_3 can not generate

$R_1 \cap R_2 = \{11(0+1)^*11, ...\} = R_3$

Set of all strings starting & ending with 11.

Q24 P-6

Sol. $\Pi_1 = (0+\lambda)(1+10)^*$

$\Pi_2 = (1+01)^*(0+\lambda)$

$\Pi_3 = (1+\lambda)(0+01)^*$

$\Pi_4 = (0+10)^*(1+\lambda)$

$\Pi_3 \rightarrow (1+\lambda)(0+01) = (0+01)(0+01) = 00$

$\Pi_4 \rightarrow (0+10)^*(1+\lambda) = (0+10)(0+10) = 00$

$\therefore \Pi_3$ & Π_4 generate 2 consecutive 0's.

$\Pi_1 \rightarrow 0(1+10)(1+10) \rightarrow 01010$ } no 2 consecutive

$\Pi_2 \rightarrow (1+01)(1+01)(0+\lambda) \rightarrow 01010$ } 0's

Q22 P-9

Sol. i) $(a+ba)^*(b+\epsilon)$

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

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

i) → ϵ ii) → ϵ iii) → does not generate ϵ Ans = a)

Q Construct RE that generate all strings of 0's and 1's where

i) length of the string exactly 4

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

ii) length of the string at least 4

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

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

iii) length of the string at most 5

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

iv) No of 1's at most 4.

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

v) length of the string divisible by 3.

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

vi) No of 0's divisible by 3

$(1^*01^*01^*01^*)^* + 1^*$

only 1's, no of 0's like 0.

Scanned by CamScanner

Starting and ending with different symbol

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

Starting and ending with same symbol

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

4th i/p symbol is 0. from R.H.S.

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

3rd i/p symbol is 1 from L.H.S.

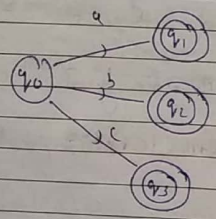
$$(0+1)(0+1)1(0+1)^*$$

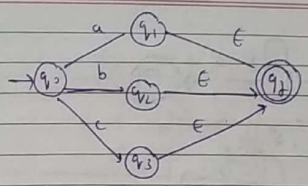
If The string starting with 0, total length is even or if the string starting with 1 total length is odd.

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

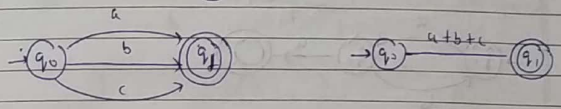
F.A to Regular Expression

1)

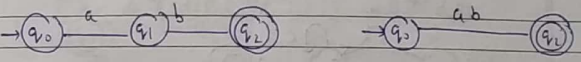




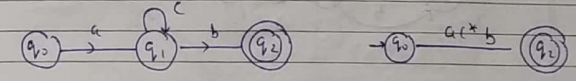
2)



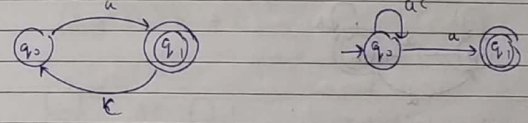
3)



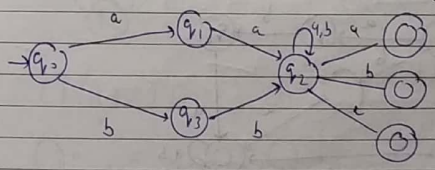
4)



5)



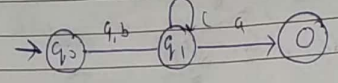
Q Construct R.E equivalent to following F.A



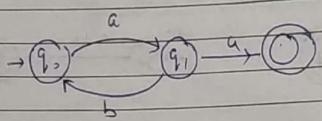
R.E :- $(aa+bb)(a+b)^*(a+b+c)$

Scanned by CamScanner

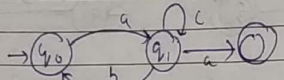
Date __/__/____ Page No.: ____

Q 

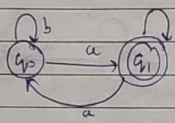
$(a+b)c^*a$

Q 

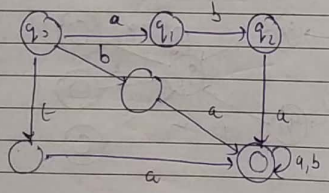
$(ab)^*aa$ (or) $a(ba)^*a$

Q 

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

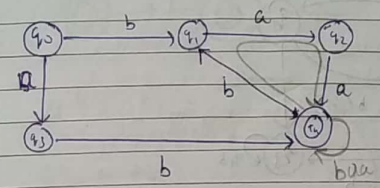
Q 

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

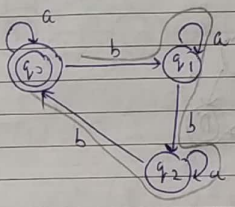
Q 

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

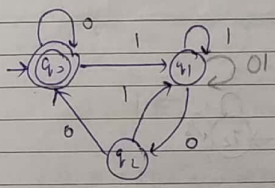
Date __/__/____ Page No.: ____

Q 

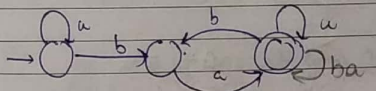
$(baa+ab)(baa)^*$

Q 

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

Q 

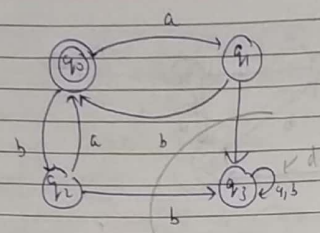
$(1(1+01)^*00+0)^*$

Q 

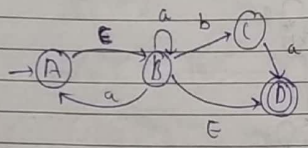
$a^*ba(a+ba)^*$

Scanned by CamScanner

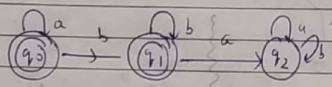
Date __/__/____ Page No.: ____

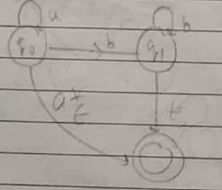
Q 

$(ab + ba)^*$

Q 

$(a^*a)^*\varepsilon.(ba + \varepsilon)$
$(a^+)^*\varepsilon.(ba + \varepsilon)$
$a^*(ba + \varepsilon)$

Q 

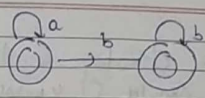


a^*bb^*

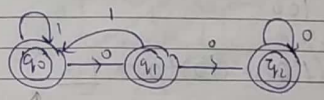
$a^* + a^*b^+$
$a^*(\epsilon + b^+)$
a^*b^*

"It is the prime responsibility of every citizen to feel that his country." —Sardar Vallabhai Patel

Date __/__/____ Page No.: ____

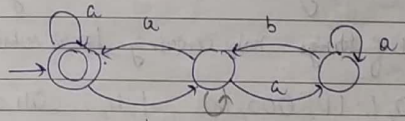


$a^* + a^*bb^*$
$a^*(\epsilon + bb^*)$
$a^*(\epsilon + b^+)$
a^*b^*

Q 

$(1+01)^* + (1+01)^*0 + (1+01)^*000^*$

$(1+01)^*(\varepsilon + 0 + 00^+)$
$(1+01)^*0^*$

Q 

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

Q42 P-9 (Gate - 1997)

Sol. RE for set of all strings not containing 100 as substring

"Education is the manifestation of perfection already in a man." —Swami Vivekananda

Scanned by CamScanner

Date __/__/____ Page No.: ____

Concept:- A R.E denoting a lang (set of strings) means it should generate all strings in L and not generate any string not in L.

a) $0^*(1^*0)^*$

generate 100

$0^*(10)(1^*0) = (10)(0) = 100$

b) $0^*10^* \rightarrow 0^*10^2 = 10100$

c) $0^*1^*01^*$

it does not generate "100" as substring

L = { set of strings not containing 100 as substring }

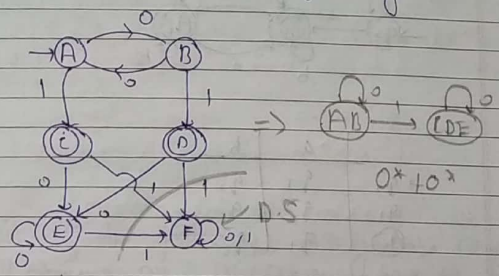
L = { E, 0, 1, 11, 000, 00 ... 1, 11, ... 011, 0111 ...

c) $0^*1^*01^*$ does not generate 1, 11, 111, ...

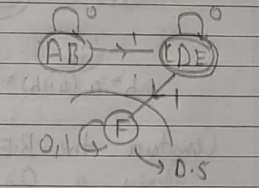
d) $0^*(10+1)^*$ ← right answer

Date 5, 10, 18 Page No.: ____

Q Construct a R.E for the following DFA



	0	1		
→ A	B	C	[AB]	[DE]
B	A	D	[AB]	[DE]
C	E	F		
D	E	F		
(E)	E	E		
F	F	F		

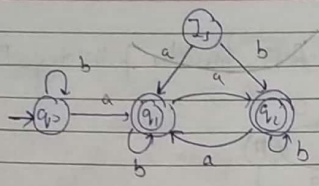


0^*10^* ← R.E

* NOTE: If possible try to minimize DFA to get RE. It is simplified form.

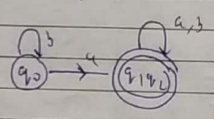
Scanned by CamScanner

Q



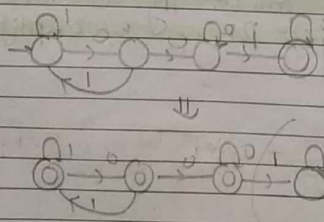
	a	b
→ q_0	q_1	q_0
q_1	q_2	q_1
(q_2)	q_1	q_2

$[q_0]$ $[q_1\ q_2]$



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

Q Construct the R.E that generates all strings of 0's & 1's where each string not having substring 001



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

$(1+01)^* + (1+01)^* 0 + (1+01)^* 000^*$

$(1+01)^* (\epsilon + 0 + 000^*)$

$(1+01)^* (\epsilon + 0(\epsilon + 00^*))$ $(\because \epsilon + RR^* = \epsilon + R^+ = R^*)$

$(1+01)^* (\epsilon + 00^*)$

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

Q Which of the following R.E generates set of all strings of 0's & 1's where no of 1's are even.

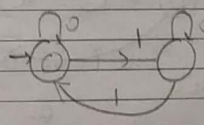
a) $0^*(10^*1)^*0^*$

b) $(010^*1)^*$

c) $(0^*(10^*1)^*)^*$

d) none

$(0^*10^*10^*)^*10^*$



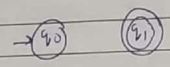
$(0+10^*1)^* \rightarrow (0^*(10^*1)^*)^*$

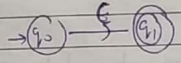
Scanned by CamScanner

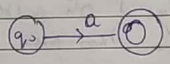
Date __/__/____ Page No.: ____

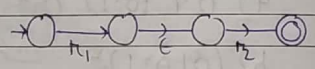
R.E to F.A (Thomson construction)

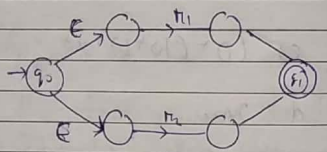
R.E ∈ NFA

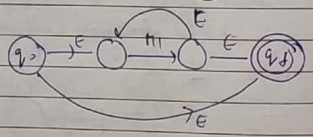
1) ϕ ⇒ 

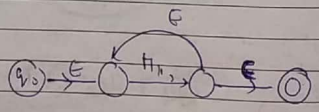
2) ϵ ⇒ 

3) a ⇒ 

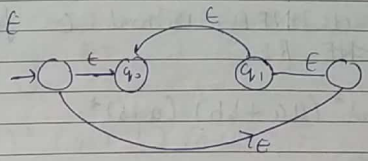
4) $r_1 . r_2$ ⇒ 

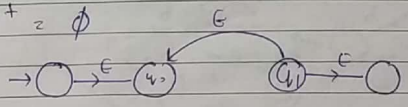
5) $r_1 + r_2$ 

6) r_1^* 

7) r_1^+ 

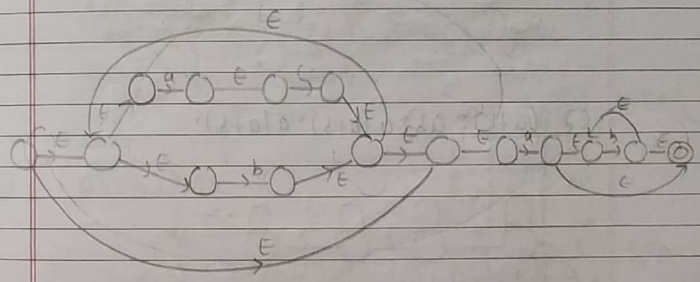
Date __/__/____ Page No.: ____

8 $\phi^* = \epsilon$ 

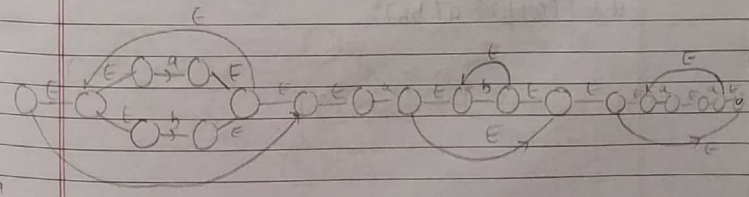
9 $\phi^+ = \phi$ 

Q Construct ∈ - NFA for the following R.E

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



ii) $(a+b)^* ab^* (aa)^*$

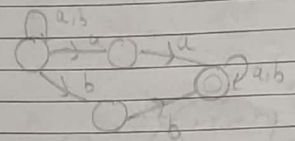


Scanned by CamScanner

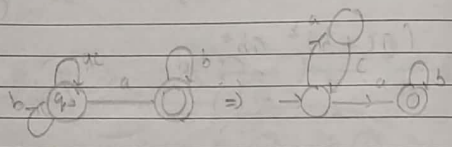
Date __/__/____ Page No.: ____

Q Construct NFA without ε for the following ~~ε-NF~~ R.E

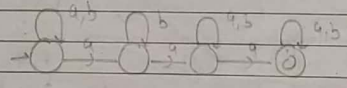
a) $(a+b)^*(aa+bb)(a+b)^*$



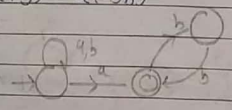
b) $(ac+b)^*ab^*$



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



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

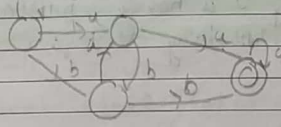


"It is the prime responsibility of every citizen to feel that his country." –Sardar Vallabhai Patel

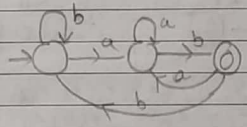
Date __/__/____ Page No.: ____

Q Construct minimal DFA ~~?~~ equivalent to following R.E

1) $(a+b)^*(aa+bb)(a+b)^*$ →



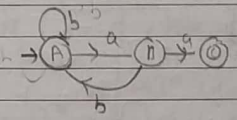
2) $(a+b)^*ab$

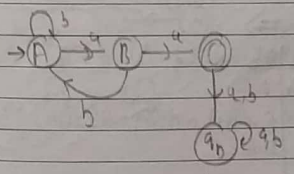


NFA

	a	b
A	B	A
B	C	A
(C)	—	—

3) $(ab+b)^*aa$





DFA

	a	b
A	B	A
B	C	A
(C)	q_D	q_D
q_D	q_D	q_D

[A B q_D] [C]
[A] [B] [q_D] [C]

"Education is the manifestation of perfection already in a man." –Swami Vivekananda

Scanned by CamScanner

Date __/__/____ Page No.:

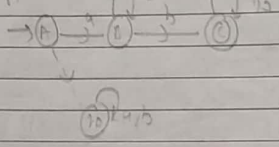
NOTE: The minimal DFA that accepts all strings of a's and b's where each string ends with particular "n" length string require "n+1" state

2) The minimal DFA that accepts all strings of a's and b's where each string is having particular "n" length substring require "n+1" states.

Ex: $(a+b)^* abab (a+b)^*$

4) $a^+b\,(a+b)^*$

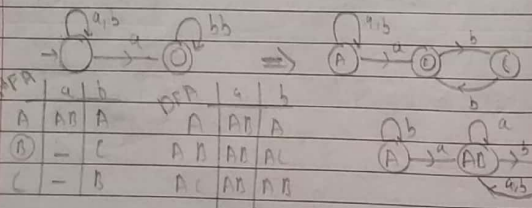
NFA



	a	b
A	B	—
B	B	C
C	C	C

	a	b
A		

5) $(a+b)^*\,a\,(bb)^*$



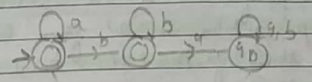
NFA	a	b
A	AB	A
(B)	—	C
C	—	B

DFA	a	b
A	AB	A
AB	AB	AC
AC	AB	AB

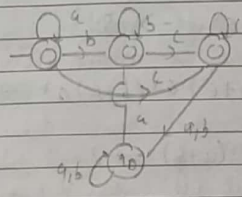
"It is the prime responsibility of every citizen to feel that his country." —Sardar Vallabhai Patel

Date __/__/____ Page No.:

6) a^*b^*



$a^*b^*c^*$



No. of states in

i) $a^*b^*c^*d^*e^* = 6$
ii) $a^*b^*c^* \ldots z^* = 27$
iii) $0^*1^*2^* \ldots 9^* = 11$

Q) find no. of states of minimal DFA equivalent to following RE

→ 3rd symbol "a" from R.H.S.

1) $(a+b)^*\,a\,(a+b)(a+b) \Rightarrow 8$

4th symbol "a" from L.H.S

2) $(a+b)(a+b)(a+b)\,a\,(a+b)^* \Rightarrow 6$

3) $(a+b)^*\,ababab \Rightarrow 7$

"Education is the manifestation of perfection already in a man." —Swami Vivekananda

Scanned by CamScanner

Date __/__/____ Page No.: ____

NOTE: Minimal DFA that accepts all strings of a's & b's where no i/p symbol a "a" from while reading the ~~As~~ string from left hand side requires "n+2" states.

4) $(a+b)^* abab (a+b)^* \rightarrow 5$

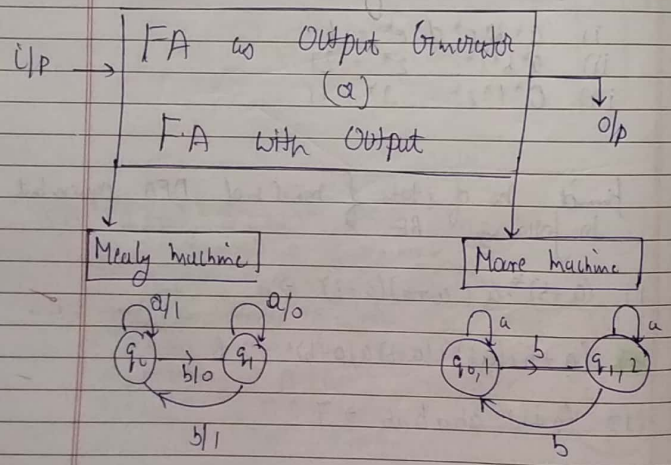
5) $a^* b a^* b a^* b a^* \rightarrow 5$

6) $a^*(b+\epsilon) a^*(b+\epsilon) a^*(b+\epsilon) a^* \rightarrow 5$

7) $((a+b)\cdot(a+b))^* (a+b) \rightarrow 2$

8) $(b^* a b^* a b^* a b^*)^* + b^* \rightarrow 3$

no q's divisible by 3



"It is the prime responsibility of every citizen to feel that his country." –Sardar Vallabhai Patel

Date __/__/____ Page No.: ____

Mealy Machine & Moore Machine

→ Mealy machine is a mathematical model in which o/p is associated with transition.

→ Moore machine is a mathematical model in which o/p is associated with state

→ Formal definition of Mealy & Moore Machine is defined with 6 tuple.

$(Q, \Sigma, q_0, \Delta, \delta, \lambda)$

Q : finite no of states.

Σ : input alphabet

q_0 : initial state

Δ : output alphabet

δ : transition function $Q \times \Sigma \rightarrow Q$

λ : Output function

Mealy machine	Moore M/c
$Q \times \Sigma \rightarrow \Delta$	$Q \rightarrow \Delta$

"Education is the manifestation of perfection already in a man." –Swami Vivekananda

Scanned by CamScanner

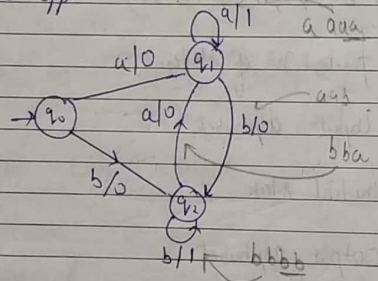
Date __/__/____ Page No.: ____

→ Both Mealy & Moore machines are deterministic in nature hence from each and every state on every i/p symbol exactly one transition should exists. (like DFA)

→ There is no final state exists in Mealy and Moore m/c.

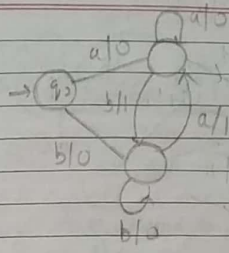
→ Mealy & Moore m/c practically used in digital circuit.

Q Construct a mealy m/c that takes all strings of a's and b's as i/p and produces output as 1 if last 2 symbols in the i/p are same. Otherwise produces 0 as o/p.

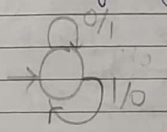


Q Construct a mealy m/c that takes all strings of a's and b's as i/p and produces 1 as o/p if last 2 symbols in the i/p are different. Otherwise produces 0 as o/p.

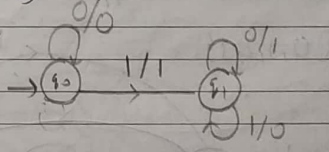
Date __/__/____ Page No.: ____



Q Construct the mealy m/c that produces 1's complement of given binary no as i/p o/p



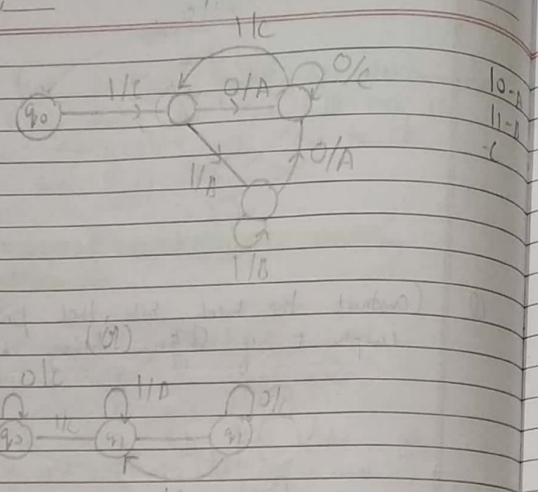
Q Construct the mealy m/c that produces 2's complement of given binary no as o/p (assume: we are reading the string from LSB to MSB)



Q Construct the mealy m/c that takes all strings of 0's and 1's as i/p and produces A as o/p if i/p ends with "10" (or) produces B as o/p if i/p ends with "11" otherwise produces C as o/p.

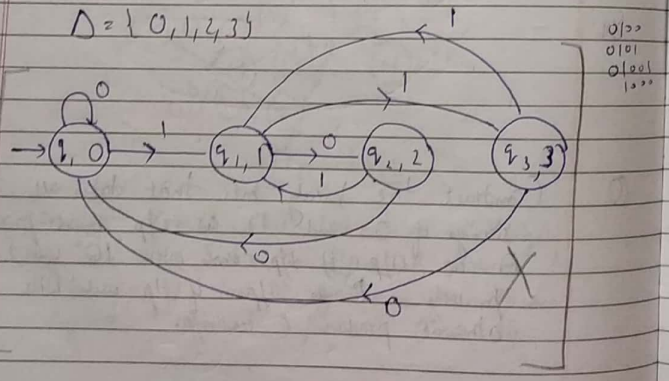
Scanned by CamScanner

Date __/__/____ Page No.: ____

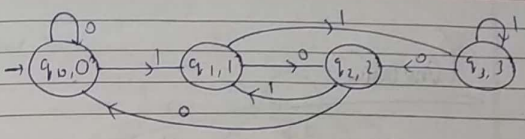


Q Construct More m/c that takes all binary no's as i/p and produces ~~residue~~ residue modulo 4 as o/p.

Sol: $\Delta = \{0, 1, 2, 3\}$

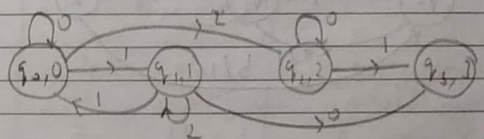


Date __/__/____ Page No.: ____



Q Construct the More m/c that takes all base 3 no's as i/p and produces residue modulo 4 as o/p.

	0	1	2
$q_{0,0}$	q_3	q_1	q_2
$q_{1,1}$	q_3	q_2	q_1
$q_{2,2}$	q_2	q_3	q_0
$q_{3,3}$	q_1	q_2	q_3



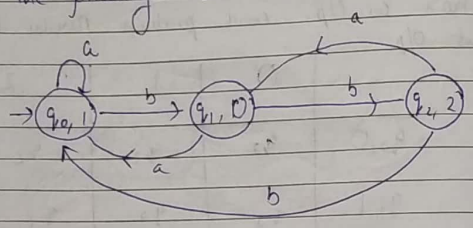
NOTE: In a More m/c that takes any base k as no's as i/p and produces residue modulo n as o/p require "n" states

Scanned by CamScanner

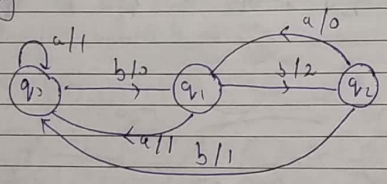
Date 6/10/18 Page No.:

Conversion

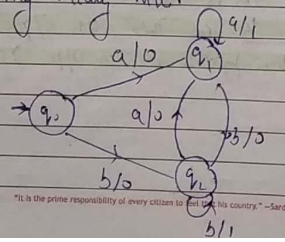
Q Construct an equivalent Mealy m/c for the following Moore m/c.



Sol: Mealy m/c

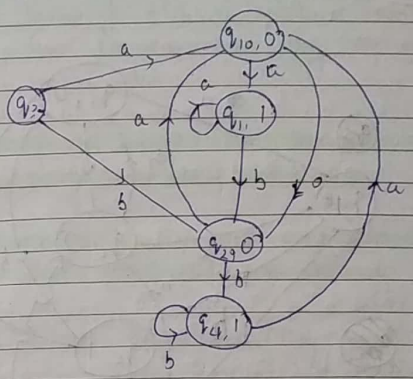


Q Construct an equivalent Moore m/c for the following Mealy m/c.



Date __/__/__ Page No.:

Sol: Moore m/c



NOTE: While converting from Mealy to Moore m/c no. of states may increase.

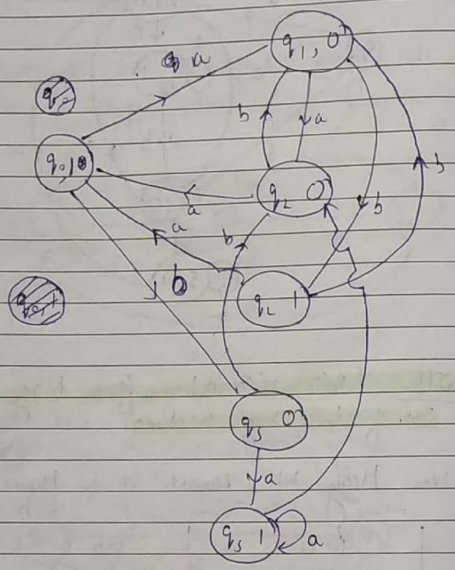
Q. given Mealy m/c convert it to Moore m/c

	i=a		i=b	
	N.S	o/p	N.S	o/p
q_0	q_1	0	q_3	0
q_1	q_2	0	q_2	1
q_2	q_0	1	q_1	0
q_3	q_3	1	q_2	0

Scanned by CamScanner

Date __/__/____ Page No.: ____

Moore m/c



State	o/p	i=a NS	i=b NS
q_0	1	q_{10}	q_{30}
q_1	0	q_{20}	q_{21}
q_{20}	0	q_{01}	q_{10}
q_{21}	1	q_{01}	q_{10}
q_{30}	0	q_{31}	q_{20}
q_{31}	1	q_{31}	q_{20}

Date __/__/____ Page No.: ____

NOTE:

While converting from Mealy m/c to Moore m/c if the initial q_0 state is splitted into multiple states, then we can take any one of that the state as initial state.

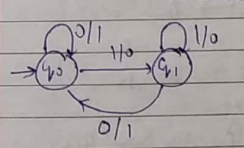
2) For a given n-length i/p sequence o/p length produced by Moore m/c is n+1.

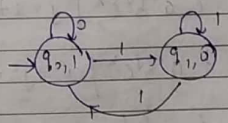
3) For the n length i/p sequence o/p length produced by Mealy m/c is n only.

4) To construct equivalent Mealy & Moore m/c for the given i/p & o/p length produced by (initial) Moore m/c is neglected.

5) For the given n-states, m-o/p symbols Mealy (m/c : maximum no. of states possible in Moore m/c is mn.

$\underbrace{m+m+m+\ldots+m}_{n \text{ times}}$ Each state can split into "m" states.





101 → 010

~~1/1/0/1~~

101 → ~~1~~ 010

neglect the initial o/p

Scanned by CamScanner

12) $(a^p)^*$ $p=2$ $(a^2)^* = \epsilon, aa, aaaa, aaaaaa, \ldots$
$p=3$ $(a^3)^* = \epsilon, a, aaa, aaaaa, \ldots$
$p=7$ $(a^7)^* = \epsilon, aaaaaaa, \ldots$
$(0, 2, 3, 4, 5, 6, 7, \ldots)$
$= (a^p)^* \Rightarrow R$
Page No.: (instead of all possible string)

Date

Regular language

Q Which of the following languages are regular

1) $L = \{a^n b^n c^n / 1 \le n \le 100\}$ → finite → Regular language

2) $L = \{a^n b^m / n+m = 100\}$ → finite → R.L

3) $L = \{a^n b^m / n-m = 5\} \Rightarrow a^{m+5} b^m$ → not R.L

4) $L = \{a^n b^m / n \times m = 100\}$ → R.L

5) $L = \{a^n b^m / n > m \text{ and } n < m\} \Rightarrow \{\} \rightarrow$ R.L

6) $L = \{a^n b^m / n \ge m \text{ and } n \le m\} \Rightarrow a^n b^n$ → not R.L

7) $L = \{a^n b^m / n = 2m+1\} \Rightarrow a^{2m+1} b^m$ → not R.L

8) $L = \{a^n b^m c^n d^m / n, m \ge 1\}$ → not R.L

9) $L = \{a^i b^j c^k / j = i+k\} \rightarrow a^i b^j b^k c^k$ → not R.L

10) $L = \{a^i b^j c^k / i^2 + j^2 = k^2\}$ → not R.L

11) $L = \{a^p / p \in \text{prime no.}\}$ not R.L

12) $L = \{(a^p)^* / p \in \text{prime no}\}$ $p = 2, 3, \ldots$ → R.L

13) $L = \{a^{2^n} / a \ge 1\}$ → not R.L c.d ≠ constant
c.d = common difference

14) $L = \{a^k / k \in \text{odd no}\}$ → R.L

15) $L = \{a^{2n+1} / n \ge 1\}$ → R.L c.d = 2

16) $L = \{x / x \in \{a,b\}^* \; n_a(x) = n_b(x) + 1\}$ not R.L

17) $L = \{x / x \in \{a,b\}^* \; n_a(x) > n_b(x)\}$ → not R.L

18) $L = \{x / x \in \{a,b\}^* \; n_a(x) \bmod 3 = 0 \text{ and } n_b(x) \bmod 5 = 0\} \rightarrow$ R.L
R.L

19) $L = \{x / x \in \{a,b\}^* \; n_a(x) \bmod 3 = n_b(x) \bmod 3\}$
R.L

20) $L = \{x / x \in \{a,b\}^* \; n_a(x) \bmod 3 > n_b(x) \bmod 3\}$
R.L

21) $L = \{x \$ y / x, y \in \{a,b\}^*\}$ → not R.L
even length palindrom over one i/p symbol

22) $L = \{ww^R / w \in \{a\}^*\}$ → R.L

23) $L = \{wbw^R / w \in \{a\}^*\}$ → not R.L
put ϵ ∈ bingmarital and b ∉ {a}*

24) $L = \{wxw^R / w \in \{a,b\}^*\}$ → not R.L

25) $L = \{wxw^R / w, x, \in \{a,b\}^*\} \rightarrow$ R.L $a(a+b)^*a + b(a+b)^*b$

26) $L = \{ww / w \in \{a,b\}^*\} \rightarrow$ R.L
$\therefore L = \{\epsilon, aa, aaaa, aaaaaa, \ldots\} = (aa)^* \rightarrow$ R.L

"It is the prime responsibility of every citizen to feel that his country." —Sardar Vallabhai Patel

"Education is the manifestation of perfection already in a man." —Swami Vivekananda

Scanned by CamScanner

27) $L = \{ ww \mid w \in \{a,b\}^* \}$ → not R.L

28) $L = \{ wxw \mid w \in \{a,b\}^* \}$ → not R.L

29) $L = \{ wxw \mid w, x \in \{a,b\}^* \}$ → R.L

30) $L = \{ wxw^R \mid w, x \in \{a,b\}^+ \}$ → R.L

31) $L = \{ ww^R x \mid w, x \in \{a,b\}^+ \}$ → not R.L

32) $L = \{ 1, 2, 4, 8 \ldots 2^n \ldots \}$ all these numbers in unary. → not R.L

33) $L = \{ ww^R ww^R \mid w \in \{a,b\}^* \}$ → R.L

34) $L = \{ a^{100^{100^{100}}} \}$ → R.L

35) $L = \{ a^n b^{2m} c^{5k} \mid n, m, k \geq 1 \}$ → R.L

36) $L = \{ a^n b^{m^2} c^{k^3} \mid n, m, k \geq 1 \}$ → not R.L

37) $L = \{ a^n b^m \mid (n+m) \bmod 2 = 0 \}$ → R.L

38) Set of all even length palindrom strings of english language → not R.L

39) Set of all even length palindrom strings of Hindi language → not R.L

40) Set of all palindrom strings of (Mandarin) Chinese lang. → not R.L

41) Set of all balanced parenthesis → not R.L

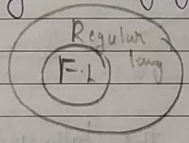
42) Equal no. of open and closed parenthesis (comparison) → not R.L
Ex:))((or (())

43) Total amount of Indian black money in Swiss Bank. R.L

44) Total number of backlog subjects of engineering students in the world. R.L

Detection of Regular Language.

1) Any finite language is always regular but all regular languages need not be finite.



2) Any infinite lang. require memory than it is non-Regular. (Comparison exists)

3) All palindrom languages formed over more than one symbol are not regular but over one symbol are regular.

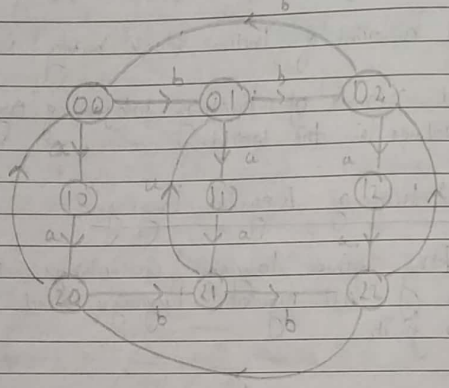
4) If any infinite language formed over one symbol, then if common difference exists b/w strings, then it is regular otherwise non-regular.

Scanned by CamScanner

Date __/__/____ Page No.:

5) By default given language is infinite language

$L = \{x \in \{a,b\}^* \mid n_a(x) \bmod 3 = n_b(x) \bmod 3\}$



Q Which of the following languages ~~are~~ is Regular.

a) $\{a^{m^n} / m \geq 1\}$

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

c) $\{a^{m^n} / n \geq 1 \cdot m > n\}$

d) none

i) $n=1$

$a^m = a^2 a^3 a^4 a^5 \ldots$

ii) $n=2$

$a^m = a^3 a^4 a^5 \ldots$

all strings will be generated except $\in$ & a^1

"It is the prime responsibility of every citizen to feel that he is country." —Sardar Vallabhai Patel

e) $\{a^{\left[\frac{m}{n}\right]} \; m,n \geq 1 \; \Delta \; n<m\}$

Date __/__/____ Page No.:

Q Which of the following not Regular.

a) $L = \{w(wR)^* / w \in \{a,b\}^*\} = (a+b)^*$

b) $L = \{w(w)^* / w \in \{a,b\}^*\}$

c) $L = \{(a^p)^* / p \in \text{prime}\}$

d) none

Sol: a) $L = w(wR)^*$

$= w(wR)^0 \Rightarrow w \in \{a,b\}^* = (a+b)^*$

Hence w generates all possible strings over ip $\{a,b\}$.

$\therefore w(wR)^*$ is R.L

b) $L = w(w)^*$

$w(w)^0 = w \in \{a,b\}^* = (a+b)^*$

Hence w generates all possible strings

$\therefore w(w)^*$ is R.L

c) $L = (a^p)^*$

put $\Rightarrow p=2$

$(a^2)^* = \epsilon, a^2, a^4, a^6, a^8, \ldots$

$(a^3)^* = \epsilon, a^1, a^3, a^5, a^7, \ldots$

$(a^5)^* = \epsilon, a^5, a^{10}, a^{15}, \ldots$

$\therefore (a^1)^*$ generates All possible strings over i/p alphabet $\{a\}$.

"Education is the manifestation of perfection already in a man." —Swami Vivekananda

Scanned by CamScanner

Date __/__/____ Page No.: ____

Pumping Lemma of Regular Lang

↳ pumping lemma is a tool used to prove a non-regular lang. as non-regular

↳ Pumping lemma is based on Pigeon Hole principle

↳ Pumping lemma uses proof by contradiction technique.

↳ Pumping lemma is strong for proving non-regular languages. & Das non-regular

↳ Pumping Lemma is weak for proving regular to lang. as regular.

To prove a lang L is non-regular by using pumping lemma.

I> Assume L is regular

II> If L is regular there exists finite automata for L having n states.

III> Select string z from L such its length |z| ≥ n

IV> According to Pidgeon Hole Principle whenever string length * is greater than ~~n~~ . or equal to no. of states.

Date __/__/____ Page No.: ____

thm at least 1 loop or cycle should exists in the automata.

V> Divide string "z" into 3 parts "u,v,w" where v is loop string.

VI> If L is regular $\forall i \geq 1$ $uv^iw \in L$

→ ~~is~~ for at least one i value.

ie $\exists i$ $uv^iw \notin L$ thm L is not regular.

By using pumping lemma to prove L is regular infinite cases to be proved.

By using pumping lemma to prove L is non-regular one case is sufficient.

Hence pumping lemma is strong to proving non-regular language.

Q There exists a regular lang. L whose finite automata is F. All strings of the language having length less than no. of states of F. Then given language is known as ____.

Ans:- finite languages

Scanned by CamScanner

Date __/__/____ Page No.:

Closure Properties of Regular Lang

① Subset Operation

Subset of a R.L is may or may not be regular hence R.L's are ~~not~~ not closed under subset op^n.

Ex: $\{a^n b^n / n \geq 1\} \subseteq \{(a+b)^*\}$

↑ not R.L

Q Which of the following is true

a) Subset of a regular set is always regular

b) Subset of a non-regular set is always regular.

c) Subset of infinite set is always regular

d) Subset of finite set is always regular.

② Concatnation Operation

Concatnation of two R.L's is always regular bcz we can construct F.A for $L_1 . L_2$ by using thomson Construction Algorithm

Date __/__/____ Page No.:

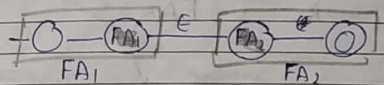
Hence R.L's are closed under Concatnation

Ex: $L_1 \{a^n / n \geq 1\}$ $L_2 \{b^n / n \geq 1\}$

$L_1 . L_2 = \{a^n\} . \{b^n\}$

↓ ↓

FA_1 FA_2



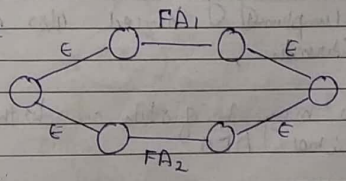
③ Union Operation

Union of two R.L's is always regular bcz we can construct F.A for $L_1 \cup L_2$ by using thomson Construction algorithm Hence R.L's are closed under Union op^n

Ex: R. L_1 — R FA_1

L_2 — R FA_2

$L_1 \cup L_2$



Scanned by CamScanner

Date __/__/____ Page No.:

(4) Complement

If L is R.L then $\Sigma^* - L$ is also regular bcz we can construct DFA complement.

Hence R.L's are closed under complement opn.

L_1 — R — FA

↓c

$\Sigma^* - L$ ⟶ FA^c

5) Intersection

Intersection of two R.L's is always Regular. bcz

L_1 ⟶ R.L

L_2 — R.L

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

Regular languages are closed under union & complement hence also closed under intersection.

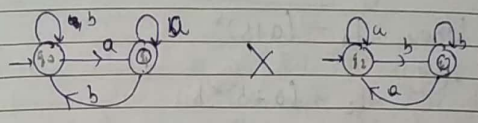
Grate

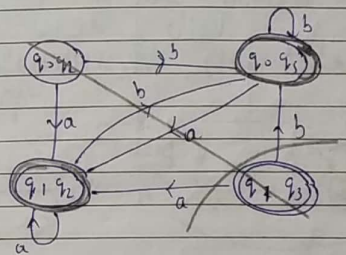
Q How many no. of states are there in $L_1 \cap L_2$ minimal DFA

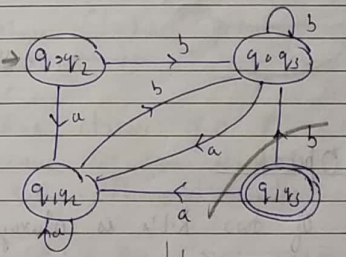
Where $L_1 = (a+b)^* a$ $L_2 = (a+b)^* b$

Date __/__/____ Page No.:

Sol: $L_1 = (a+b)^* a$ $L_2 = (a+b)^* b$







F.A

inaccessible

$q_1 q_3 = A$

$q_2 q_3 = C$

$q_0 q_3 = B$

$q_0 q_2 = A$

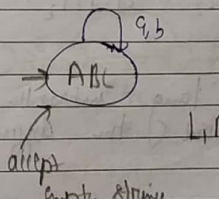
	a	b
A	C	B
B	C	B
C	C	B
D	C	B

[ABC] [D]

[ABC]

Minimize

$\{q_0 q_2 \; q_0 q_3 \; q_1 q_2\}$ $\{q_1 q_3\}$



accept empty string

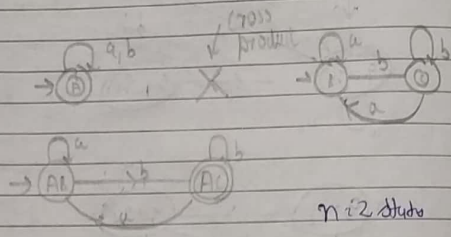
$L_1 \cap L_2 = \{\}$ empty language

Q How many states are there in $L_1 \cap L_2$.

Where $L_1 = (a+b)^*$

$L_2 = (a+b)^* b$

Sol $L_1 = (a+b)^*$ $L_2 = (a+b)^* b$



n = 2 states

⑥ Difference Operation

Difference of two R.L's is always Regular. bcz

$L_1 - L_2 = L_1 \cap L_2^C$

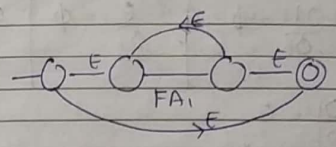
In any lang intersection and complement is closed then difference also closed.

7) Klem closure Operation

Klem closure of a R.L is always Regular because we can construct F.A for L^* by using Thomsons construction

Hence R.L's are closed under klem closure Op^n.

$L_1^* =$

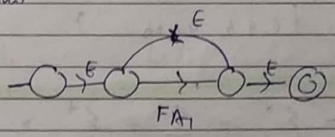


8) Positive Closure Operation

Positive closure of a R.L is always regular because we can construct F.A for R.L L^+ by using Thomsons construction

Hence R.L's are closed under Positive Closure operation.

$L_1^+ =$



Scanned by CamScanner

Date / /

Page No.:

Quotient Operation

$L_1/L_2 = \{ x \mid xy \in L_1 \; y \in L_2 \}$ → right cancellation

$L_1 = xy = x$ | $\frac{101}{01} = 1$ | $\frac{010}{00} = \phi$

$L_2 = y$ | $\frac{011}{01} = \phi$ | $\frac{\epsilon}{01} = \phi$ | $\frac{01}{\epsilon} = 01$

$\frac{01}{01} = \epsilon$

$\frac{0^*}{0} = \{ \frac{\epsilon}{0}, \frac{0}{0}, \frac{00}{0}, \frac{000}{0} \ldots \} = \{ \phi, \epsilon, 0, 00, 000, \ldots \} = 0^*$

$\frac{0^*1}{1^*0} = \phi$

$\frac{\Sigma^*}{\Sigma^*} = \{ \frac{\epsilon}{\epsilon}, \frac{a}{\epsilon}, \frac{b}{\epsilon}, \frac{aa}{\epsilon}, \frac{bb}{\epsilon} \ldots \} = \Sigma^*$

$\{ \epsilon, a, b, aa, bb \ldots \}$

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

"To manifest the divinity is the right of everyone." —Sri Ramakrishna Paramahamsa

Date / /

Page No.:

Q) $L_1 = a^*ba^*$

$L_2 = b^*a$

find $L_1/L_2 = ?$

Approach by method of elimination, first try to eliminate ϵ if possible.

a) $a^*ba^* + a^*$

b) a^* → this generates only a's, try to check whether $\frac{L_1}{L_2}$ can generate b also.

c) a^*ba^*

d) $a^*b + a^+$

$L_1 = ba$ $L_2 = ba$

$\frac{L_1}{L_2} = \epsilon$ c) ✗ d) ✓

$L_1 = aba$ $L_2 = a$

$\frac{L_1}{L_2} = \frac{aba}{a} = ab$ b) ✗

Q) $L_1 = a^*b$ $L_1/L_2 = ?$

$L_2 = b^*$

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

c) a^*b d) b^*

$L_1 = b$ $L_1 = ab$

$L_2 = b$ $L_2 = b$

$\frac{L_1}{L_2} = \epsilon$ $\frac{L_1}{L_2} = a$

a) ✗ d) ✗

c) ✗

"No real change in history has ever been achieved by discussions." —Subhash chandra bose

Scanned by CamScanner

Date __/__/____ Page No.:

NOTE: Quotient of two R.L is always regular bcz we can construct FA of L_1/L_2 as follows

Quotient FA Construction

→ $\frac{L_1}{L_2}$ FA is L_1 F.A only but there is a difference in final states.

→ To decide whether 1st state is final or not starting from the 1st state consider the total automata. If resultant automata accepts any string of L_2 then make 1st state as final otherwise non-final.

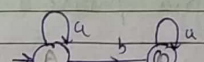
→ To decide 2nd state final (or) not, starting from 2nd state consider the remaining automata. if this automata accepts any string of L_2 then make 2nd state as final otherwise non-final.

→ Repeat this process for every state to decide whether a state is final (or) not.

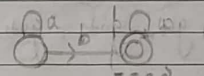
Date __/__/____ Page No.:

Ex: $L_1 = a^*ba^*$ $L_2 = b^*a$

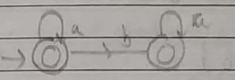
L_1/L_2 = Quotient

L_1 = 

accepts ba of L_2 ∴ A is final state



accepts a of L_2 ∴ B is final state



$a^* + a^*ba^*$

Init (or) Prefix Operation

Init of a language is set of all prefixes of all strings of given language

Prefix :- sequence of leading symbols of the given string

TOC & gate

ε, T, TO, TOC & ε, g, ga, gat, gate

Scanned by CamScanner

Date __/__/____ Page No.:

→ For n-length i/p string total no. of prefixes possible is n+1

→ For n length i/p string total no. of proper prefixes possible is n

Ex: gate

ϵ, g, ga, gat, gate ← prefixes

ϵ, g, ga, gat ← proper prefixes.

$L_1 = \{ab, ba\}$

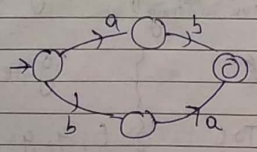
Init(L) = {ϵ, a, ab, b, ba}

$L_2 = \{x \mid x \in \{a,b\}^* \quad n_a(x) = n_b(x)\}$

$= \{\epsilon, a, b, ab, ba, aa, \ldots\}$

$= (a+b)^*$

$L_1 = \{ab, ba\}$

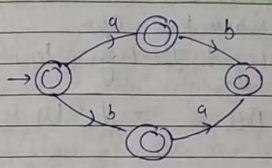


Date __/__/____ Page No.:

NOTE:

Init of a R.L is always regular bcz we can construct F.A for init(L) by making all states as final except dead state.

init(L) = {ϵ, a, ab, b, ba}



11) Infinite Union

$\{L_1 \cup L_2 \cup L_3 \ldots\} \quad \forall i \; L_i \to R.L$

→ Infinite Union of regular lang's may (or) may not be regular.
(are not)

→ Hence R.L's are closed under infinite union operation

Ex: $\{ab\} \cup \{aabb\} \cup \{aaabbb\} \cup \ldots = \{a^n b^n \mid n \geq 1\}$
↑ not R.L

12) Infinite Intersection

$\{L_1 \cap L_2 \cap L_3 \cap L_4 \ldots\} \quad \forall i \; L_i \to R.L$

Scanned by CamScanner

Date __/__/____ Page No.: ____

Infinite intersection of R.L's is may (or) may not Regular.

Hence R.L's are not closed under infinite intersection opn according to De-Morgan's law

$L_1 \cap L_2 \cap L_3 \cap L_4 \ldots = L_1' \cup L_2' \cup L_3' \ldots$

(R R R)

NOTE:

R.L's are closed under finite union & finite intersection but not closed under infinite union & infinite intersection.

Q) Which of the following opn are closed under R.L's?

a) Union b) intersection

c) both a) & b) d) none of this

NOTE:

By default union means finite union and intersection means finite intersection

Date __/__/____ Page No.: ____

3) Substitution Operation

$\Sigma \rightarrow \Delta$

Substitution is a mapping from $(\Sigma \rightarrow \Delta)$ i.e alphabet Σ to alphabet Δ where each symbol of Σ is replaced by a regular language over the alphabet Δ

Substitution function is applied as follows

1) $S(\phi) = \phi$

2) $S(\epsilon) = \epsilon$

3) $S(a+b) = S(a) + S(b)$

4) $S(a \cdot b) = S(a) \cdot S(b)$

5) $S(a^*) = (S(a))^*$

NOTE:

Substitution of a R.L is always regular bcz we can construct FA for substituted language by performing substitution fun^n directly on FA/automata transitions.

$\Sigma = \{0\}$ $\Delta = \{a, b\}$

$S(0) = (a+b)(a+b)$ ← exactly 2 length strings

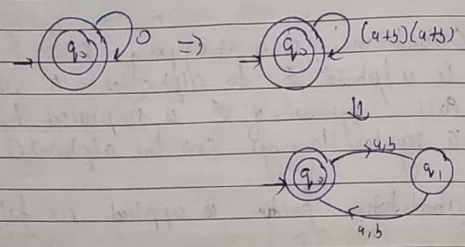
Scanned by CamScanner

Date __/__/____ Page No.: ____

length of string changes

$L = 0^*$

$S(L) = S(0^*) = (S(0))^* = ((a+b)(a+b))^*$



4) Homomorphism

Homomorphism is a special case of substitution where each symbol of alphabet Σ is replaced by single string over the alphabet Δ.

Homomorphism of a regular lang is always regular bcz we can construct FA similar to substitution.

Hence R.L's are closed under homomorphism operations

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

$h(0) = aa$
$h(1) = \epsilon$

"To manifest the divinity is the right of everyone." –Sri Ramakrishna Paramahamsa

Date __/__/____ Page No.: ____

$L = 1^*01^*$

$h(L) = h(1^*01^*)$

$= h(1^*) \cdot h(0) \cdot h(1^*)$
$= (h(1))^* \cdot h(0) \cdot (h(1))^*$
$= \epsilon^* \cdot aa \cdot \epsilon^*$
$= aa$

$h(0) = aa$ $h(1) = bb$
$L = (ab+ba)^*$
$= h(ab) + h(ba)$
$h(L) = h((ab+ba)^*)$

$= (h(ab+ba))^*$ $(h(ab)+h(ba))^*$

$h(a) = 0$ $h(b) = 1$ $h(c) = 10$

$L = \{101\}$ $h(L)$

$h(L) = h(101)$

$= \{$

$= (h(ab)+h(ba))^*$
$= \{$

"His real change in history has ever been achieved by discussions." –Subhash Chandra Bose

Scanned by CamScanner

Date __/__/____ Page No.:

5) Inverse Homomorphism Operation

Applying Homorphism in reverse way is called inverse homomorphism
i.e string is replaced by symbol

NOTE: $h(h^{-1}(L))$ may or may not be L

~~$\Sigma = \{a, b, c\}$~~

Inverse Homorphism of a regular language is always regular.

Hence RL is closed under inverse homorphism op^n

$\Sigma = \{a, b, c\}$ $\quad \Delta = \{0, 1\}$

$h(a) = 0 \quad h(b) = 1 \quad h(c) = 10$

$L = \{10, 01\}$

$h^{-1}(L) = \{cc, cba, baba, bac\}$

$h(h^{-1}(L)) = \{10, 10, 10, 10\}$

$= \{10\}$

Date __/__/____ Page No.:

$h(0) = aa \quad h(1) = bb$

$L = (ab + ba)^*$

$h^{-1}(L) = \phi^* = \epsilon$

$h(h^{-1}(L)) = h(\epsilon) = \epsilon$

6) Reversal Opn

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

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

→ Reverse of a regular language is always regular Hence RL's are closed under Reversal opn

→ Reversal Automata Construction

i) Interchange initial state as final state and final states as initial state

ii) Reverse transition direction.

iii) If multiple initial states exist in Reversal Automata convert them into single initial (by taking new initial state and attaching with ϵ transitions
↓
State

Scanned by CamScanner

Date __/__/____ Page No.:

iii) Reversal opn performed on R.E as follows.

1) $(E+F)^R = E^R + F^R$

2) $(E.F)^R = F^R . E^R$

3) $(E^*)^R = (E^R)^*$

Q. Which of the following language not necessary regular where L is regular language

a) $\{uv \mid u \in L, v \in L^R\}$

b) $\Sigma^* - L$

c) $\{ww^R \mid w \in \{a,b\}^*\}$ → ww length palindrom.

d) none

Gate

Imp NOTE:

If $L_1 \cap L_2$ is regular and L_1 is regular L_2 need not be necessarily regular.

Ex: $L_1 = \{ab\} \cap L_2 = \{a^n b^n / n \geq 1\} = \{ab\}$

L_1: R, L_2: Not R, $L_1 \cap L_2$: R

Date __/__/____ Page No.:

Q. $L_1 \cup L_2$ is regular & L_1 is regular then L_2 is

a) should be regular

b) need not be regular

Ex. $L_1 = (a+b)^*$ (R) $\cup$ $L_2 = \{a^n b^n / n \geq 1\}$ (not R) $= (a+b)^*$ (R)

Q. $L_1 . L_2$ is regular & L_1 is regular

a) should be regular

b) need not be regular.

Ex. ϕ. $\{a^n b^n / n \geq 1\} \Rightarrow \phi$.

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

(X: R, ww^R: Not R)

where $X = (a+b)^*$

Imp Q. Consider the following two statements

S1: In Any NFA if all states are final, that NFA accepts Complete language.

Scanned by CamScanner

S_2: There exists a Regular language A for all other regular languages B. such that $A \cap B$ is Regular

a) S_1 & S_2 are True

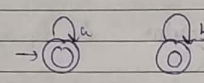
b) S_1 & S_2 are False

c) S_1 is true S_2 is False

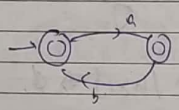
d) S_1 is false, S_2 is true

Sol:

S_1:



(or)



S_2: $A \cap B = Reg.$

$\epsilon \cap B = \epsilon$ reg

(or)

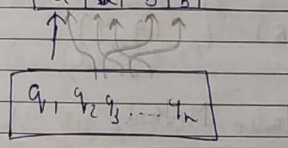
$\phi \cap B = \phi$ reg

Decision Properties of Regular Language

Following problems are decidable under Regular languages.

i) Membership Problem
ii) ~~Emptiness~~ Finiteness Problem
iii) Emptiness problem
iv) Equivalence Problem

2 DFA (or) Two-way DFA



→ DFA capable to move left as well as right side direction known as 2 DFA.

→ Formal definition of 2 DFA is :-

Scanned by CamScanner

Page No.:

Date

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

$\delta: Q \times \Sigma \to Q \times \{L, R\}$

→ Expressive power of 2DFA is same as DFA. Hence 2DFA also accepts R.L only.

Q Check whether the string "1010011" is member of following 2DFA (or) not

	0	1
→ q_0	(q_0, R)	(q_1, R)
(q_1)	(q_1, R)	(q_2, L)
q_2	(q_0, R)	(q_2, L)

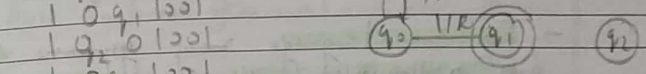
Sol:- (Accept) Acceptance Method

By reading any string 2DFA enters into final state, then that string is accepted otherwise string is rejected.

Date 7/10/18

Page No.:

q_0 1 0 0 1
1 q_1 0 0 1
1 0 q_1 0 1
1 q_2 0 0 1
1 0 q_0 0 1
1 0 1 q_1 0 1
1 0 1 0 q_1 1
1 0 1 0 0 q_1
1 0 1 0 q_2 1
1 0 1 q_0 0 1
1 0 1 0 0 q_1
1 0 1 0 0 1 (q_1) ← F



Grammar

Set of rules used to describe strings of a language is known as Grammar.

Formal definition of Grammar is:

$G = (N, T, P, S)$

N: Set of non-terminals (or) variables
T: set of terminals
P: no of productions
S: Starting Symbol

→ For every language Grammar exists

↳ Every grammar generates one language

Scanned by CamScanner

$\alpha \rightarrow \beta$ means α gives β (or) α derives β (or) α finds to β

Date __/__/____ Page No.:

↳ Every grammer is of the form $\alpha \rightarrow \beta$. Where α & β are equal.

→ For one language many grammers there is a possibility of having more than one grammer

Derivation

Process of deriving strings from the given grammer is known as derivation.

Derivation can be Left Most Derivation (or) Right Most Derivation

Parse Tree (or) Derivation Tree

Tree representation of derivation is known as parse tree.

All leaf nodes of parse tree is known as yield of the parse tree.

Sentential form

Every step in the derivation is known as Sentential form

If the derivation is Left most derivation then Sentential form is known as Left Sentential form

Date __/__/____ Page No.:

If the derivation is Right Most Derivation then the Sentential form is known as Right Sentential form

→ For every language Grammer exists. hence for Regular language Grammer exists known as Type-3 grammer.

→ Grammar for CFL is known as Type-2 grammer

→ Grammer for CSL is known as Type-1 grammer

→ Grammer for REL is known as Type-0 (or) Unrestricted Grammer.

Left Most Derivation (LMD)

It is a derivation in which left most non-terminal is replaced by its right hand side part at every step.

Right Most Derivation (RMD)

It is a derivation in which Right most non terminal is replaced by its right hand side part at every step.

Scanned by CamScanner

Date / /

Page No.:

Q. Construct grammer following language

1) $L_1 = \{a^n b^m \mid n, m \geq 1\}$

$S \to AB$
$A \to aA \mid a$
$B \to bB \mid b$

2) $L_2 = \{a^n b^m \mid (n+m)$ is even$\}$

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

$S \to AB \mid aAB \; aAbB$
$A \to aaA \mid \epsilon$
$B \to bbB \mid \epsilon$

3) $L_3 = \{a^n b^m \mid (n+m)$ is odd$\}$

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

$S \to AbB \mid aAB$
$A \to aaA \mid \epsilon$
$B \to bbB \mid \epsilon$

4) $L_4 = \{a^n b^n \mid n \geq 0\}$

$S \to aSb \mid \epsilon$

$L = \{\epsilon, ab, aabb, \ldots\}$

"To manifest the divinity is the right of everyone." –Sri Ramakrishna Paramahamsa

Date / /

Page No.:

5) $L_5 = \{a^n b^n c^m d^m \mid n \geq 0 \; m \geq 0\}$

$S \to AB$
$A \to aAb \mid \epsilon$
$B \to cBd \mid \epsilon$

6) $L_6 = \{a^n b^n c^m d^m \mid n \geq 0 \; m \geq 1\}$

$S \to AB$
$A \to aAb \mid \epsilon$
$B \to cBd \mid cd$

7) $L_7 = \{a^n b^m c^m d^n \mid n, m \geq 1\}$

$S \to aSd \mid aAd$
$A \to bAc \mid bc$

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

$L = \{a^n a^m b^m c^m\}$

$S \to aSc \mid aAc$
$A \to aAb \mid ab$

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

$L = \{a^n b^n b^m c^m\}$

$S \to AB$
$A \to aAb \mid ab$
$B \to bBc \mid bc$

"No real change in history has ever been achieved by discussions." –Subhash chandra bose

Scanned by CamScanner

Date __/__/____ Page No.:

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

$S \rightarrow aSbb / abb$

Q Find the grammar that generates all strings of a's and b's, where

i) Each string starting and ending with different symbol.

ii) Each string contains substring aaa.

iii) Each length of the string exactly 4.

iv) length of the string divisible by 2

v) length of the string odd.

vi) Each string is even length palindrom.

v) Each string is odd length palindrom

Sol: i) $a(a+b)^*b + b(a+b)^*a$

$S \rightarrow aAb / bAa$

$A \rightarrow aA / bA / \epsilon$

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

$S \rightarrow AaaaA$

$A \rightarrow aA / bA / a / b / \epsilon$

"To manifest the divinity is the right of everyone" –Sri Ramakrishna Paramahamsa

Date __/__/____ Page No.:

iii) $S \rightarrow AAAA$

$A \rightarrow a / b$

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

$S \rightarrow AS / \epsilon$

$A \rightarrow BB$

$B \rightarrow a / b$

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

$S \rightarrow AB$

$A \rightarrow BBA / \epsilon$

$B \rightarrow a / b$

vi) $S \rightarrow aSa / bSb / \epsilon$

vii) $S \rightarrow aSa / bSb / a / b$

Q Construct a grammar that generates set of all odd length palindrom strings of English language

$S \rightarrow aSa / bSb / cSc / \ldots zSz / a / b / c / \ldots / z$

Q Construct the grammar that generates set of all odd length palindrome strings of Hindi language

$S \rightarrow$ अ S अ / आ S आ / ... ज्ञ S ज्ञ / अ / आ / ... / ज्ञ

"No real change in history has ever been achieved by discussions." –Subhash chandra bose

Scanned by CamScanner

Date __/__/____ Page No.:

Q Declaration stmt grammer

int a;
int a,b,c;

S → TD
T → int | float | char
D → id , D | id

Q Grammer for expression

a = b + c × d

S → id = E
E → E+E | E×E | id

Q Identify language generated by following grammar

i) S → aS | bS | a

ii) S → aSbS | bSaS | ϵ

iii) S → aAa | bAb | a | b
A → aA | bA | ϵ

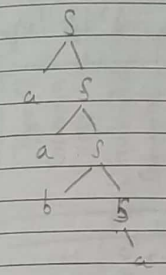
iv) S → aSb | aAb
A → aA | a

v) S → aSb | aAb | aBb
A → aA | a
B → bB | b

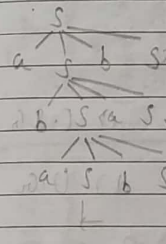
"To manifest the divinity is the right of everyone" –Sri Ramakrishna Paramahamsa

Date __/__/____ Page No.:

i) S → (a+b)*a



ii) S → aSbS | bSaS | ϵ



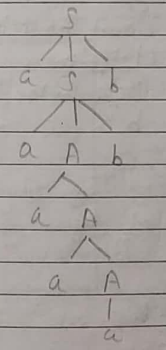
$L = \{ x \mid x \in \{a,b\}^* \}$
$n_a(x) = n_b(x)$

iii) a(a+b)*a + b(a+b)*b + a + b

iv) S → aSb | aAb
A → aA | a

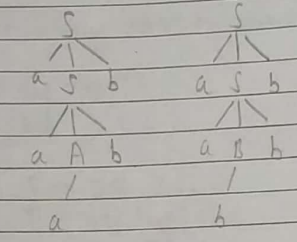
$a^n b^m \mid n > m \quad n,m \geq 1$

$a^m A b^m$
$a^m a^+ b^m$
$a^n b^m \mid n > m$
$n,m \geq 1$



"No real change in history has ever been achieved by discussions." –Subhash chandra bose

Scanned by CamScanner

v) 

$a^n b^m \mid n \neq m$

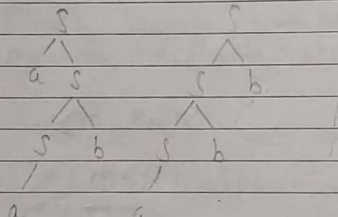
$n, m \geq 1$

Following Grammar is

$G : S \to aS \mid Sb \mid a$

a) Regular Grammar c) C.S.G

b) CFG d) U.G



G can not be RG bcz for G to be RG, it must be either completely Left linear or completely right linear, but not both at same time

"To manifest the divinity is the right of everyone" —Sri Ramakrishna Paramahamsa

Types of Grammars (Chomsky)

- Type 0 Grammar (OR) Unrestricted Grammar (OR) RE G
  - $\alpha \to \beta$
  - $\alpha \in (V+T)^+$
  - $\beta \in (V+T)^*$
- Type 1 Grammar (OR) Context Sensitive Grammar
  - $\alpha \to \beta$
  - $|\alpha| \leq |\beta|$
  - $\alpha, \beta \in (V+T)^+$
- Type 2 Grammar (OR) Context Free Grammar
  - $A \to \alpha$
  - $\alpha \in (V+T)^*$
- Type 3 Grammar (OR) Regular Grammar
  - Left linear Grammar: $A \to Bx \mid x$, $x \in \Sigma^*$
  - Right linear Grammar: $A \to xB \mid x$, $x \in \Sigma^*$

"No real change in history has ever been achieved by discussions." —Subhash chandra bose

Scanned by CamScanner

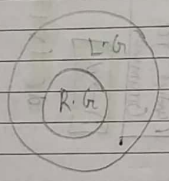
Date __/__/____ Page No.:

→ All Grammers is of the form $\alpha \to \beta$ where α should contains at least one non-terminal.

Linear Grammar (L.G)

In ~~Linear~~ any grammar L.H.S one non-terminal exists and R.H.S at most one non-terminal exists is known as Linear grammar

Ex:- $S \to aSb/ab$
$S \to aS/Sb/a$



L.G can be Left linear or right linear or middle linear

If any Linear Grammar is either Left Linear or right linear (but not both) is R.G

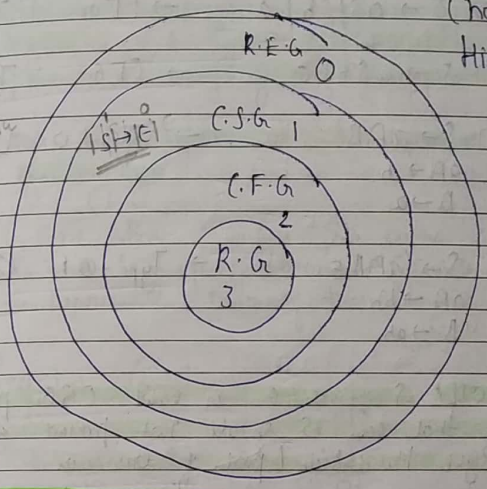
Every R.G is L.G, but every L.G need not be Regular

Date __/__/____ Page No.:

following grammars are linear but not Regular

Ex: $S \to aSb/ab$,
$S \to aS/Sb/a$

(Chomsky Hierarchy)



to cover ε production only when S will present at R.H.S

$S \to \varepsilon$

Type i+1 ⊂ Type i i = 0, 1, 2

3 ⊂ 2, 1, 0
2 ⊂ 1, 0
1 ⊂ 0

Scanned by CamScanner

NOTE: always start checking from type 3 grammer.

Date __/__/____ Page No.:

Q Identify types of following grammers

1) $S \to aS/bS/a$ → R.G Type-3

2) $S \to aS/Sb/a$ → CFG Type-2

3) $S \to aSb/\epsilon$ → CFG Type-2

4) $S \to aAB$ → Type-0 U.G
$aA \to b$
$B \to a$

5) $S \to aAB/\epsilon$ → Type-1 CSG
$aA \to bb$
$B \to ab$

NOTE: S gives ϵ is valid CSG production in that case S should not present in Right Hand Side part of Grammer.

6) $S \to aAB/\epsilon$ → Type-0 U.G
$aA \to abS$
$B \to b$

7) $S \to \epsilon/aS/bS$ → Type-3 R.G

↳ Regular grammars are suitable to represent rules of Regular language only (no comparision exists) for

"To manifest the divinity is the right of everyone." –Sri Ramakrishna Paramahamsa

Date __/__/____ Page No.:

→ To represent rules of non-regular languages context free grammars (or) context sensitive grammars are used.

→ We can not use it for semantics

→ CFGs are used to represent syntax of languages, By using CFGs we can not represent meanings of programming statements.

→ CSG's are used to represent semantics of language.

→ C, C++, Java languages are called as Context Sensitive languages because CSG's are used to represent rules of these languages.

Regular Grammar

→ R.G generates R.L only.

→ For every R.G, R.E can be constructed.

→ For every R.G, F.A can be constructed. and vice-versa.

→ For every Right linear grammar, left linear grammar can be constructed. and vice-versa.

→ For every F.A, Right linear grammar can be constructed.

"No real change in history has ever been acheived by discussions." –Subhash chandra bose

Scanned by CamScanner

Date __/__/____ Page No.:

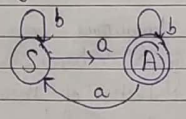
→ For every F.A, left linear grammar can be constructed.

→ NOTE: i)

i) By default Regular Grammar means Right Linear Grammar.

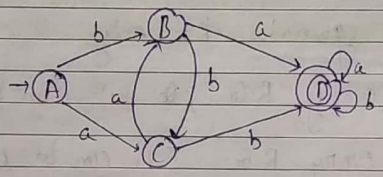
Q Construct Regular Grammar Equivalent to following F.A

i)



$S \rightarrow bS \mid aA$
$A \rightarrow aS \mid bA \mid \epsilon$

ii)



$A \rightarrow bB \mid aC$
$B \rightarrow bC \mid aD$
$C \rightarrow bD \mid aB$
$D \rightarrow aD \mid bD \mid \epsilon$

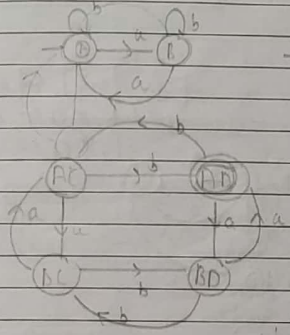
Date __/__/____ Page No.:

Q Construct the R.G that generates all strings of a's and b's where

i) no. of a's even and no. b's odd.

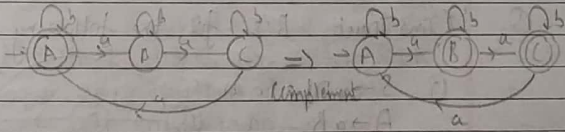
ii) no of a's not divisible by 3.

Sol' i)



$AC \rightarrow aA \mid bAD$
$AD \rightarrow aBD \mid bAC \mid \epsilon$
$BC \rightarrow aAC \mid bBD$
$BD \rightarrow aAD \mid bB$

ii)



$A \rightarrow aB \mid bA$
$B \rightarrow aC \mid bB \mid \epsilon$
$C \rightarrow aA \mid bC \mid \epsilon$

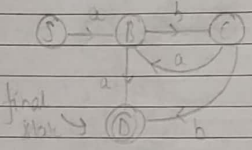
Scanned by CamScanner

Date __/__/____ Page No.: ____

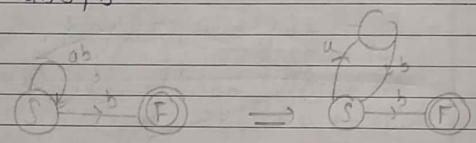
NOTE: FA to R.G construction algorithm always results Right linear grammar

Q Construct the F.A equivalent to following R.G

1) $S \to aB$
$B \to bC \mid a$
$C \to aB$
$C \to b$

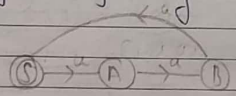


Q. 2) $S \to abS \mid b$



Q Construct R.E for the following R.G

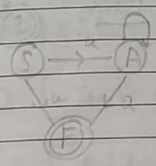
1) $S \to aA \mid \epsilon$
$A \to aB$
$B \to aS$



$(aaa)^*$

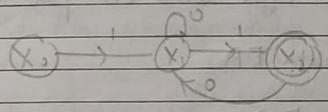
Date __/__/____ Page No.: ____

2) $S \to aA \mid a$
$A \to aA \mid bA \mid a$



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

3) $X_2 \to 1X_1$
$X_1 \to 0X_1 \mid 1X_2$
$X_2 \to 0X_1 \mid \epsilon$



$1(0+10)^*1$

Q Construct F.A for the following Left linear Grammar

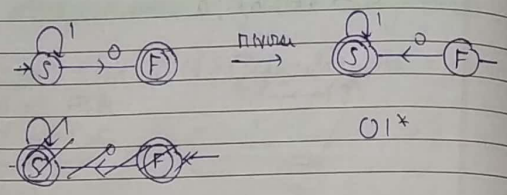
$S \to S1 \mid 0$

Steps → 1 → Reverse Gr. to R.L.G
2 → Construct FA
3 → Reverse FA

Scanned by CamScanner

Date __/__/____ Page No.: ____

$S \to 1S|0$



01^*

Construction of F.A from L.L.G

1) Reverse Right Hand side part of Grammar
2) Construct F.A
3) Reverse F.A

Construction of R.L.G from F.A

1) Reverse F.A
2) Construct R.L.G.
3) Reverse R.H.S part of Grammar

Create

Identify language generated by following Regular Grammar?

Date __/__/____ Page No.: ____

$S \to aA|bB$
$A \to a\,B|bC|b$
$B \to a\,C|bA|a$
$C \to aA|bB|a$

a) $L = \{ x / x \in \{a,b\}^* \; n_a(x) = 3n_b(x) \}$

b) $L = \{ x / x \in \{a,b\}^* \; n_a(x) = n_b(x) + 3 \}$

c) $L = \{ x / x \in \{a,b\}^* \; 3n_a(x) = n_b(x) \}$

d) none

Sol:- Concept: RG generates only R.L

Context Free Grammar

Regularity Problem

NOTE: Regularity problem of C.F.G is undecidable problem.

To check whether given C.F.G generates regular language (or) not. is undecidable problem (no algorithm exists)

Ambiguity Problem

next page

Scanned by CamScanner

Date __/__/____ Page No.: ____

Q Which of the following grammar's generates regular language

a) $S \to AB$
$A \to aA/\epsilon$
$B \to bB/\epsilon$

b) $S \to aA$
$A \to aAB/\epsilon$
$B \to c$

c) $S \to AB$
$A \to aAb/\epsilon$
$B \to bB/\epsilon$

d) None

Sol a) $a^n b^m \mid n,m \geq 0$

b) $a^n c^m \mid n > m$

c) $a^n b^m \mid n < m$

Ambiguity Problem (Imp) (asked 9 times)

A grammar is said to be ambiguous; for at least one string x, if there exists more than one L.M.D (Left Most Derivation) (or) more than one R.M.D (Right Most Derivation) (or) more than one Parse Tree.

Unambiguous Grammar

A grammar is said to be unambiguous for all strings of the grammer only one parse tree exists (or) only one L.M.D exists (or) only one R.M.D exists.

Date __/__/____ Page No.: ____

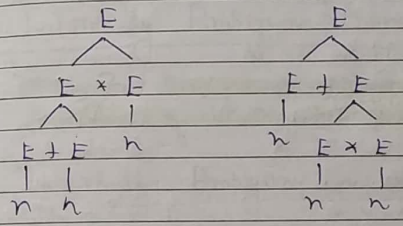
Q Which of the following grammar is ambiguous

① $E \to E+E/E \times E/n$

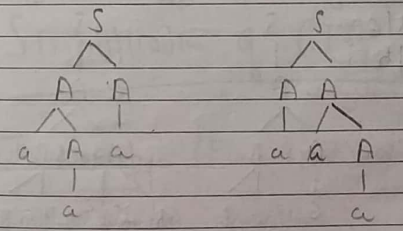
② $S \to AA$
$A \to aA/a$

③ $S \to aSbS/\ bSaS/\epsilon$

Sol 1) $n+n \times n$ Ambiguous



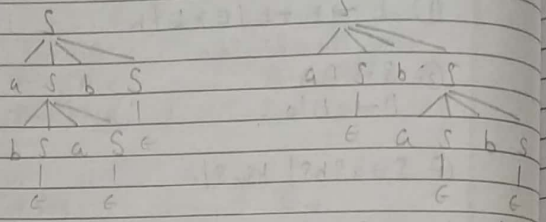
2) $S \to AA$ aaa Ambiguous
$A \to aA/a$



Scanned by CamScanner

3) $S \to aSbS \mid bSaS \mid \epsilon$ Ambiguous

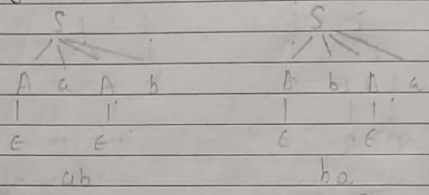
abab



4) $S \to AaAb \mid BbBa$ Unambiguous

$A \to \epsilon$

$B \to \epsilon$

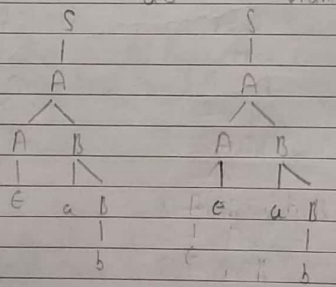


ab ba

5) $S \to A$ ab Unambiguous

$A \to AB \mid \epsilon$

$B \to aB \mid b$



NOTE: i) Ambiguity is undecidable problem because there is no algorithm exists to check grammar G is ambiguous (or) not.

ii) Elimination of Ambiguity is undecidable problem hence there is no algorithm exists to eliminate ambiguity from any ambiguous grammar.

Inherently Ambiguous Grammar

- It is a ambiguous grammar from which elimination of ambiguity is not possible

Inherently Ambiguous language means :-

A language is said to inherently ambiguous if all grammars possible for that language are inherently ambiguous.

Simplification of CFG

To simplify CFG we need to eliminate

1) Useless variables
2) Unit - production
3) Null production

Scanned by CamScanner

Date __/__/____ Page No.: ____

Elimination of Useless Variable :

It means eliminate variables which are not generating any string. and eliminate variables which are not required for derivation.

Q Eliminate Useless Variables from the following grammar productions.

1) $S \to AB/a$
$A \to BC/b$
$B \to bC/Ba$
$C \to aB/bC$
$D \to a$

2) $S \to AB/AC$
$A \to aAb/bAa/a$
$B \to bbA/aaB/AB$
$C \to abCa/aDb$
$D \to bD/aC$
$E \to a$

3) $S \to A/B$
$F \to 1$
$H \to 1/\epsilon$
$D \to \epsilon$
$A \to EA/FC$
$B \to EB/FD$
$E \to OEF/OFG$
$F \to OEF/OFH$

4) $S \to AB/a$
$A \to a$
$B \to bB$
$C \to a$

Sol: 1) $D \to a$
$B \to bC/Ba$
$C \to aB/bC$
$A \to BC$
$S \to AB$

$S \to a$

2) $E \to a$
$D \to bD/aC$
$C \to abCa/aDb$

$A \to aAb/bAa/a$
$B \to bbA/aaB/AB$
$S \to AB$

4) $C \to a$
$B \to bB$
$S \to AB$

$S \to a$

3) $A \to FC$
$E \to OEF/OFG$
$F \to OEF$
$A \to EA$
$B \to EB$
$S \to A$

$S \to B$
$F \to 1$
$H \to 1/\epsilon$
$D \to \epsilon$
$B \to FD$
$E \to OFH$

Unit Production

Any production is of the form $A \to B$ is known as unit production.

If unit productions present in the grammar then difficult to detect useless variable, hence to simplify the grammar first eliminate unit production then useless variable.

Q Eliminate unit production and useless variable from the following grammar.

1) $S \to A$
$A \to B$
$B \to C$
$C \to D$
$D \to a$

2) $S \to Aa/B$
$B \to A/bb$
$A \to a/bc/B.$

"To manifest the divinity is the right of everyone" –Sri Ramakrishna Paramahamsa

"No real change in history has ever been acheived by discussions." –Subhash chandra bose

Scanned by CamScanner

Date / /

Page No.:

1) D → a
C → aD
Replace D by a
C → a
B → a
A → a
S → a

2) A → a | bc | bb
B → a | bc | bb
S → Aa | a | bc | bb

S → Aa | B
replace by RHS of B

Elimination of Null Productions

To simplify the grammar first eliminate Null productions and then Unit productions and then Useless Variable.

Q Eliminate Null productions from the following grammars

1) S → aS, b
S, → aS, b | ∈

Sol: S → aS, b | ab
S, → aS, b | ab

2) S → aA | bS
A → bS | aA | ∈

S → aA | bS | a
A → bS | aA | a

3) S → aAB
A → BC
B → a | ∈
C → b | ∈

C → b
B → a
A → BC | C | B
S → aAB | aA | aB | a

4) S → AbB
A → BC
B → a | ∈
C → b | ∈

B → a
C → b
A → BC | C | B | ∈
S → AbB | bB | Ab | b

Unit Production Eliminate

A → BC | b | a
S → AbB | bB | Ab | b

Normal form of CFG

Applying condition on R.H.S of grammar is known as Normal Form of the CFG

There are two normal forms in C.F.G's. They are :-

i) Chomsky Normal Form → A → BC, A → a | a ∈ T
ii) Greibach Normal Form → A → aα | α ∈ V*

To normalize the grammar into CNF (or) GNF it should be simplified

Q Convert following CFG into CNF grammar,

Scanned by CamScanner

Date / /

Page No.:

$S \to bA \mid aB$
$A \to bAA \mid aS \mid a$
$B \to aBB \mid bS \mid b$

1) Let $X = a$ $Y = b$

$S \to YA \mid XB$
$A \to YAA \mid XS \mid a$
$B \to XBB \mid YS \mid b$

2) Let $P \to AA$ $Q \to BB$

$S \to YA \mid XB$
$A \to YP \mid XS \mid a$
$B \to XQ \mid YS \mid b$
$P \to AA$
$Q \to BB$

Steps:-

Convert Grammar R.H.S parts into only non-terminals by assuming terminal symbols as new non-terminal.

Multiple non-terminals on the R.H.S, convert into only 2 non-terminals by introducing new non-terminals.

2) $S \to aAB \mid BB$
$A \to a$
$B \to b$

"To manifest the divinity is the right of everyone" –Sri Ramakrishna Paramahamsa

Date / /

Page No.

Sol 1) Let $b = Y$ $a = X$

$S \to XAB \mid BY$
$A \to a$
$B \to b$

2) Let $P \to AB$

$S \to XP \mid BY$
$A \to a$
$B \to b$
$P \to AB$

3) $S \to aSb \mid ab$

Let $S \to b$. & Let $X \to a$ $Y \to b$

$S \to XSY \mid XY$.

Let SY $P \to SY$

$S \to XP \mid XY$
$P \to SY$
$X \to a$
$Y \to b$

Greek

NOTE: To generate n-length string from CNF Grammar, total no. of productions required is "2n-1" (ie) no of derivation steps)

(or)

2) To generate "n" length string from CNF grammar total no. of productions required is "n" only

n = no of derivation steps.

"Any real change in history has ever been achieved by discussions." –Subhash Chandra Bose

Scanned by CamScanner

Date __/__/____ Page No.:

Q Convert following CFG into GNF grammar.

1) $S \to aSb \mid ab$

Steps:

1) Convert total R.H.S into only non-terminals
2) First non-terminal on the R.H.S is replaced by its R.H.S

1) Let $A \to a$, $B \to b$
$S \to ASB \mid AB$

2) $S \to AB$
$A \to aA \mid bB \mid a$
$B \to b$

Let $P \to a$ $Q \to b$ ✗

$S \to AB$
$A \to PA \mid QB \mid a$

$S \to aAB \mid bBB \mid aB$
$B \to b$
$A \to aA \mid bB \mid a$.

Date __/__/____ Page No.:

3) $S \to Sa \mid b$

1) Let $P \to a$, $S \to SP \mid b$ 2) $S \to bP \mid b$, $P \to a$

Let $A \to a$ $S \to SA \mid b$
$SaA \mid bA \mid b$ $B \to b$
$SAA \mid BA \mid b$
$SaAA \mid bAA \mid bA \mid b$

NOTE: Left recursive grammars can not be converted into GNF.
By eliminate left recursion from grammar we can construct GNF grammars.

Q Eliminate Left Recursion from the following grammar productions

$S \to Sa \mid b$

$S \to bS'$
$S' \to aS' \mid \epsilon$

Concept: $A \to A\alpha \mid \beta$

$A \to \beta A'$
$A' \to \alpha A' \mid \epsilon$

Scanned by CamScanner

Date / / Page No.:

Generalize

$A \to A\alpha_1 / A\alpha_2 / A\alpha_3 / \dots A\alpha_n / \beta_1 / \beta_2 / \beta_3 / \dots \beta_m$

~~$A \to \beta_1 / \beta$~~

$A \to \beta_1 A' / \beta_2 A' / \beta_3 A' / \dots / \beta_m A'$

$A' \to \alpha_1 A' / \alpha_2 A' / \alpha_3 A' / \dots / \alpha_n A' / \varepsilon$

4) $E \to E+T / T$
$T \to T*F / F$
$F \to a$

$T \to FT'$ $\quad E \to TE'$
$T' \to *FT' / \varepsilon$ $\quad E' \to +TE' / \varepsilon$

(Imp) **Indirect Left Recursion** (Imp)

Q Eliminate indirect left recursion from the following

$A \to Ba$
$B \to Ab / a$

$A \to Ba$
$B \to BAb / a$
$B \to Bab / a$

$B \to aB'$
$B' \to abB' / \varepsilon$

"To manifest the divinity is the right of everyone." –Sri Ramakrishna Paramahamsa

Date / / Page No.:

Steps:

i) Start examining the productions one by one. Start symbol production First. Once verified remember it.

ii) While verifying the next production, if verified non-terminal appears, then substitute R.H.S of that verified non terminal in the current production which we are verifying

iii) If left recursion found then eliminate it.

Q Which of the following grammar is free from indirect recursion

a) $S \to Aa / B$
$A \to Bb / Sc / a$
$B \to a$

b) $S \to Aa / Bb / a$
$A \to Bd / a$
$B \to Aa / b$

c) $S \to Ab / Bb / c$
$A \to Bd / a$
$B \to b$

d) none

Sol: c) $S \to Ab / Bb / c$ ✓ verified — no indirect recursion
$A \to Bd / a$
$B \to b$

b) $S \to Aa / Bb / a$
$A \to Bd / a$
$B \to Bda / b$
↺ left recursion

"No real change in history has ever been acheived by discussions." –Subhash chandra bose

Scanned by CamScanner

Date __/__/____ Page No.:

Q Convert following LR grammar into GNF grammar.

$S \to AA|0$
$A \to SS|1$

Sol:

$S \to AA|0$
$A \to AS|OS|1$ ⇒
$A \to OSA'|1A'$
$A' \to ASA'|\epsilon$

$S \to AA|0$
$A \to OSA'|1A'$
$A' \to ASA'|\epsilon$

Simply the CFG

$A' \to ASA'|AS$
$A \to OSA'|1A'|OS|1$
$S \to AA|0$

↓ Convert to GNF (1st non-terminal is replaced by its R.H.S)

$S \to OSA'A|OSA|1A'A|1A|0$
$A \to OSA'a|1A'|1$
$A' \to OSA'A'|OSSA'|1A'SA'|1SA'$
$OSA'S|OSS|1A'S|1S$

"To manifest the divinity is the right of everyone" –Sri Ramakrishna Paramahamsa

Date 8/10/18 Page No.:

Decision Properties of CFG

1) Membership Problem.

→ means checking whether string x is generated from given grammar or not.

→ checking whether string x generated from grammar or not known as parsing.

→ Membership problem is decidable because there exists algorithm known as CYK algorithm.

→ CYK algorithm is also known as parsing algorithm.

→ Time Complexity of CYK algorithm is $O(n^3)$.

→ To apply CYK algorithm given grammar should be in Chomsky Normal Form (CNF)

Substring

Sequence Consequitive sequence of symbols over the given string is called substring.

Note: NOTE: Total # of strings possible for n length string is $\frac{n(n+1)}{2} + 1$ (0 length string)

"No real change in history has ever been acheived by discussions." –Subhash chandra bose

Scanned by CamScanner

Date __/__/____ Page No.: ____

Ex-1) TOC

0-1m	∈	1	
1-1m	T, O, C	3	(1+2+3)+1
2-1m	TO, OC	2	
3-1m	TOC	1	

ii) gate

0-1m	∈	1
1-1m	g, a, t, e	4
2-1m	ga, at, te	3
3-1m	gat, ate	2
4-1m	gate	1

iii) bhopal

0-1m	∈
1-1m	b, h, o, p, a, l
2-1m	bh, ho, op, pa, al
3-1m	bho, hop, opa, pal
4-1m	bhop, hopa, opal

Q1 Check whether the string "baaba" member of following grammar or not.

Yes its member

S → AB | BC
A → BA | a
B → CC | b
C → AB | a.

"To manifest the divinity is the right of everyone" –Sri Ramakrishna Paramahamsa

Date __/__/____ Page No.: ____

NOTE:
First check whether given grammar is in CNF or not. if not then convert to CNF.

Q2 Which of the following strings are members of given grammar.

a) ba b) baa c) ab d) aaba e) aba

Q3 How many substrings of the string "baaba" are members of given grammar 5
no of boxes having S

Q4 How many substrings of the string "baaba" are not members ? 11

Q5 How many different substrings of the string "baaba" are members of given grammar? 4

Q6 How many substrings of the string "baaba" are generated from "B" only ? 5

Q7 How many different substrings of the string "baaba" are generated from "B" only ? 4

Sol-1 Yes "baaba" is member of given grammar.

Sol-2) ba, ab, aaba are members of the given grammar

Sol-3) Concept: # of boxes having S = # of substrings of "baaba".
To get the substring, from the box containing "S" move diagonally up.

"No real change in history has ever been achieved by discussions." –Subhash chandra bose

Scanned by CamScanner

Date __/__/____ Page No.: ____

	0	1	2	3	4
	b	a	a	b	a
0	B	A, C	A, C	B	A, C
1	A, S	B	S, C	A, S	
2	–	B	B		
3	–	S, A, C			
4	A, S, C				

A, S, C ← last box contain S (i.e. starting symbol) given string accepted.

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

10 ⇒ B × (A, C)
BA, BC

11 ⇒ (A, C) × (A, C)
AA, AC, CA, CC

12 ⇒ (A, C) × B
AB, CB

13 ⇒ B × (A, C)
BA, BC

20 ⇒ 00 × 11 | 10 × 02
B × B | (A, S) × (A, C)
BB | AA, AC, SA, SC

AS, CS, BB
AA, AS, CA, CS
ϕ, AS, SS
BA, BS
BA

2) take substring to be cross 2 length diagonal. if box contain starting symbol S, then it is member

5) # different substrings member
Starting from box S, cross the diagonal and get the corresponding string. remove redundant string.

Date __/__/____ Page No.: ____

S.1.3) # of boxes having S = no of substrings which are members

S.1.4) no of substrings not member = total − member
= 16 − 5 (including ϵ also, 0 length string)
= 11

S.1.6) # of substrings generated by B = # of boxes containing B.

S.1.7) b, aa, aab, aba ← different substring,

<u>NOTE</u>: Filling the table

i) While filling the 1st row see where corresponding string symbol is present in grammar at R.H.S and fill the box with corresponding L.H.S non-terminal.

ii) While filling 2nd row.

ex:- filling 10 Here we need to consider
i) Cross product of 00 × 01 i.e (B × (A, C))
(BA, BC) look at R.H.S and fill corresponding L.H.S

iii) While filling 3rd row

ex:- 20 Here we need to consider
i) Cross product 00 × 11 i.e (B × B)
ii) Cross product 10 × 02 i.e (A, S) × (A, C)

S.1.1) Concept:- If last box contain S, then given string belong to given grammar.

Scanned by CamScanner

Date __/__/____ Page No.:

Q i) Check whether the string "aabbb" is member of following grammar or not

$S \rightarrow AB$
$A \rightarrow BB|a$
$B \rightarrow AB|b$

ii) Which of the following strings are members of given grammar?

a) aa b) aab c) ab d) bbb

iii) How many different substrings of the given string "aabbb" are members of given grammar.

iv) How many substrings of the given string "aabbb" are not members of given grammar.

v) How many different substrings of the given string "aabbb" are generated from "A" only.

Sol: i) Yes v) 4

ii) b, c, d

iii) 5

iv) 11

"To manifest the divinity is the right of everyone." –Sri Ramakrishna Paramahamsa

Date __/__/____ Page No.:

a	a	b	b	b
A	A	B	B	B
	S, B	A	A	
S, B	A	S, B		
A	S, B			
S, B				

Finiteness Problem

means checking whether language generated by given grammar is finite (or) not.

Procedure:

NOTE: Finiteness problem is decidable for CFG.

Procedure:

1) Simplify the grammar
2) Convert the grammar into CNF grammar
3) Construct CNF graph
4) If CNF graph contains loop (or) cycle then grammar generates infinite language

"No real change in history has ever been achieved by discussions." –Subhash chandra bose

Scanned by CamScanner

Otherwise grammar generate finite language.

→ If Language generate by given grammar is finite then rank of each variable can be defined.

→ If rank of a variable is "n" then maximum length string generated by that variable is "2^n".

→ Rank of a variable is length of the longest path starting from that variable in CNF graph.

Q Find rank of each non-terminal for the following grammar and also check given grammar generates finite language or not.

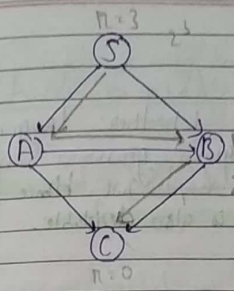
$S \to AB$
$A \to BC \mid a$
$B \to CC \mid d$
$C \to a$

Yes it accepts finite lang.

Sol:- Given grammar is already simplified and in CNF form.

Now we construct CNF graph

Start from A and go till that non-terminal (i.e.) which has 0 out degree

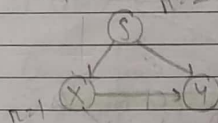


No loop or cycle
Given grammar accepts finite language.

Q Find rank of each non-terminal for the following grammar.

$S \to XY$
$X \to AX \mid YY \mid a$
$A \to BC$
$B \to AC$ → useless
$C \to AB$
$Y \to b$.

Sol: Simplified grammar $S \to XY$
$X \to YY \mid a$
$Y \to b$



Scanned by CamScanner

Date __/__/____ Page No.: ____

Emptiness Problem

means checking whether language generated by given grammar is empty (or) ~~non empty~~ not. Hence For CFG, Emptiness problem is also decidable.

Procedure:

1) Eliminate useless variables from the following grammar.

2) If starting symbol is useless, given grammar generates Empty language otherwise non-empty

Ex:- 1) $S \to aS / Sb / aSb$ S is useless.

$S \to XY$
$X \to aX$ $X \to aX$ is useless
$Y \to b$ Hence $S \to XY$ is useless

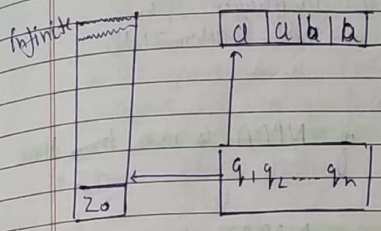
Equivalence Problem

Equivalence Problem of CFG is ~~undecidable~~ undecidable ie to check given 2 CFG's generates same language (or) not. there is no algorithm exists.

"To manifest the divinity is the right of everyone." –Sri Ramakrishna Paramahamsa

Date __/__/____ Page No.: ____

Pushdown Automata (P.D.A)



Stack + F.A = PDA

→ F.A fails to accept languages which requires infinite memory.

→ F.A having one stack as infinite memory elements is known as PDA.

→ Formal definition of PDA is:

$PDA = (Q, \Sigma, q_0, F, \delta, Z_0, \Gamma)$

Q : finite no of states
Σ : i/p alphabet
q_0 : initial state
F : set of final states
Z_0 : initial stack element
Γ : stack alphabet
δ : transition function

$$Q \times \Sigma \cup \{\epsilon\} \times \Gamma \to Q \times \Gamma^*$$

"No real change in history has ever been achieved by discussions." –Subhash chandra bose

Scanned by CamScanner

→ There is only 1 type of PDA, known as language recognizer

→ PDA can recognize languages, in deterministic way or non-deterministic way.

→ Expressive power of NPDA is more than DPDA

→ By default PDA is NPDA.

→ Language accepted by DPDA is are known as Deterministic CFL's.

→ Language accepted by ~~NDPDA~~ NPDA are known as non-deterministic CFL's.

→ PDA can be represented using

i) Transition diagram
(or)
ii) Transition notations

→ PDA practically used in compilers to design parser.

→ There are 2 types of acceptance methods in PDA, known as

i) Empty stack ii) Final state

→ Empty stack Method

by reading the complete string from left to right, end of the string, stack of the PDA is empty then given string is accepted and ~~irrelevant~~ irrelevant about no. of final state

→ Final ~~stack~~ Method
State

By reading the complete string from left to right by end of the string, PDA enters into final state then given string is accepted and irrelevant about stack empty or not

→ Language accepted by PDA is known as context free language (CFL's).

→ Drawback of PDA

i) PDA fails to accept languages which requires more than one stack

Gate Q. Let n_1 is no. of language accepted by PDA using Empty stack method.

Let n_2 is no. of language accepted by PDA by final state method.

Which of the following is true?

Scanned by CamScanner

a) $n_1 = n_2$ b) $n_1 < n_2$ c) $n_1 > n_2$

d) We can't say

NOTE:

1) No of language accepted by Empty stack method and final state method is same in PDA.

2) No of language accepted by final state method is more than Empty stack method in DPDA

Gate Q

Size of the stack in PDA is restricted to 10,000 elements only. Then language accepted by that PDA is

a) Regular language b) finite language only

c) CFL by not regular d) Regular but not CFL

NOTE: If size of the stack in PDA is restricted or stack is not used in PDA, language accepted by that PDA is Regular language. (Finite and infinite)

2) Hence PDA can recognize R.L as well as CFL's.

3) Expressive power of PDA > FA

4) The language for which PDA not possible are not CFL's.

Q Construct PDA for the language

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

$Q \times \Sigma \cup \{\epsilon\} \times \Gamma \rightarrow Q \times \Gamma$

$\delta(q_0, z_0, a) \rightarrow = (q_0, az_0)$ } push all a's
$\delta(q_0, a, a) \rightarrow = (q_0, aa)$

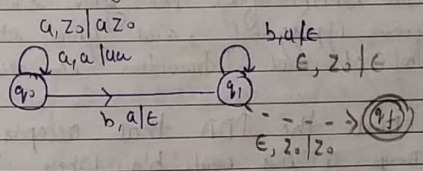
$\delta(q_0, b, a) = (q_1, \epsilon)$ } pop all a's
$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$ (or) (q_f, z_0) } acceptance

(q_1, ϵ) ↑ acceptance by empty stack

(q_f, z_0) ↑ acceptance by final state

Transition diagram



Scanned by CamScanner

Date __/__/____ Page No.:

NOTE:

1) By reading the i/p string PDA always halts.

2) Whenever PDA halts, if the string is ended and stack is empty, then given string is accepted by Empty stack method.

3) Whenever the PDA halts, if the string is not ended, then the given string is rejected.

4) Whenever the PDA halts if the string is ended and automata enters into final state, then given string is accepted.

5) Whenever PDA halts if the stack is not empty and automata is in non-final state then string is rejected.

6) In DPDA from the given state, given i/p symbol, and given stack symbol at most 1 transition can exists.

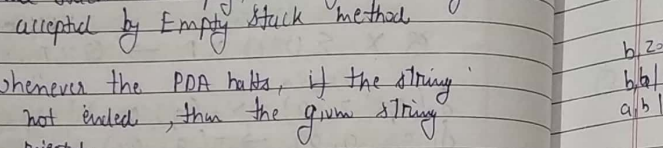
Q Construct the PDA that accepts all strings of a's and b's where no of a's and no. of b's are same. ~~in~~ ~~every step.~~

Date __/__/____ Page No.:

$L = \{ x \mid x \in \{a,b\}^* \ n_a(x) = n_b(x) \}$

$\delta(q_0, z_0, a) = (q_0, az_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_0, \epsilon)$
$\delta(q_0, \epsilon, z_0) = (q_1, \epsilon)$ — acceptance by Empty stack
(or) (q_1, z_0) ← acceptance by final state

} starting with 'a'



$\delta(q_0, z_0, b) = (q_0, bz_0)$
$\delta(q_0, b, b) = (q_0, bb)$
$\delta(q_0, a, b) = (q_0, \epsilon)$
$\delta(q_0, \epsilon, z_0) = (q_1, \epsilon)$
(or) (q_1, z_0)

} starting with 'b'

Concept:- if same symbol push, if different symbol pop

Q Construct PDA for the language that ~~accepts~~ contains all strings of a's and b's where $n_a > n_b$.

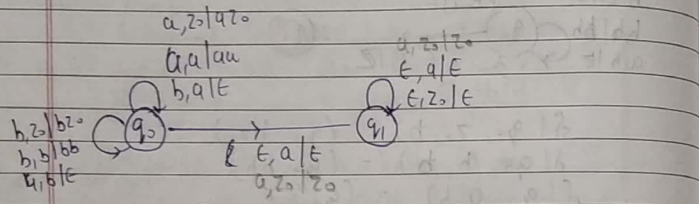
$L = \{ x \mid x \in \{a,b\}^*, \ n_a(x) > n_b(x) \}$

using Empty stack method?

Sol:

Scanned by CamScanner

$\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_0, \epsilon)$
$\delta(q_0, b, z_0) = (q_0, bz_0)$
$\delta(q_0, b, b) = (q_0, bb)$
$\delta(q_0, a, b) = (q_0, \epsilon)$
$\delta(q_0, \epsilon, a) = (q_1, \epsilon)$ $\delta(q_1, \epsilon, a) = (q_1, \epsilon)$
$\delta(q_0, \epsilon, z_0) = (q_1, \epsilon)$



Q Construct PDA for the language

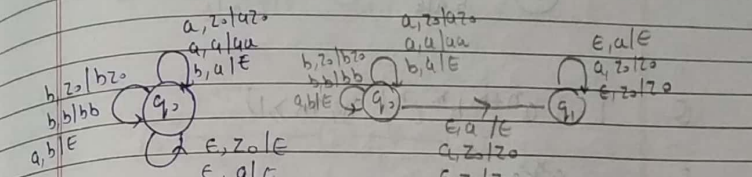
$L = \{ x / x \in \{a,b\}^* \quad n_a(x) \geq n_b(x) \}$

Sol:

$\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_0, \epsilon)$
$\delta(q_0, b, z_0) = (q_0, bz_0)$
$\delta(q_0, b, b) = (q_0, bb)$
$\delta(q_0, a, b) = (q_0, \epsilon)$

$\delta(q_0, \epsilon, a) = (q_0, \epsilon)$
~~$\delta(q_1, \epsilon, a) = (q_1, \epsilon)$~~
~~$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$~~

$\delta(q_0, \epsilon, z_0) = \delta(q_0, \epsilon)$

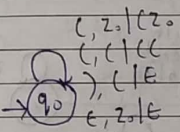


Q Construct PDA that accepts set of all balanced parenthesis

Sol:

$\delta(q_0, \epsilon, z_0) = \delta(q_0, (z_0)$
$\delta(q_0, (, () = \delta(q_0, ((\,)$
$\delta(q_0,), () = \delta(q_0, \epsilon)$
$\delta(q_0, \epsilon, z_0) = \delta(q_0, \epsilon)$

Ex: ((()))(())



Q $L = \{ a^n b^m / \quad n > m \quad , n, m \geq 1 \}$

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

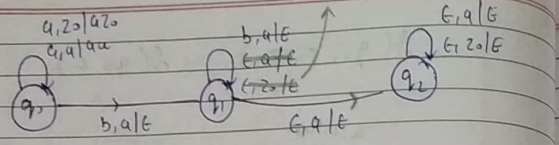
$\delta(q_1, \epsilon, a) = \delta(q_1, \epsilon)$ $\delta(q_1, \epsilon, a) = \delta(q_1, \epsilon)$
$\delta(q_1, \epsilon, z_0) = \delta(q_1, \epsilon)$

Scanned by CamScanner

Date / /

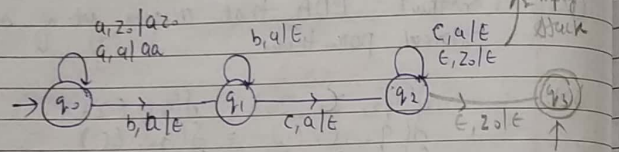
Page No.:

will accept $n_a(x) = n_b(x)$



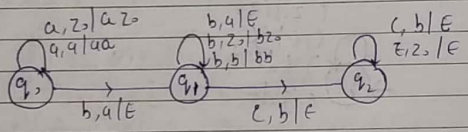
Q $L = \{ a^{n+m} b^m c^n \mid n, m \geq 1 \}$

Sol: $a^n a$



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

Sol:



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

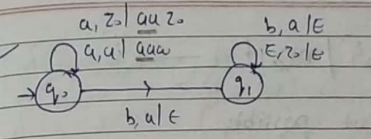
Sol: Logic 1 → for 1 a, push 2 a's and pop 1 a for every b.

Logic 2 → for 2 b, pop 1 a
ie for 1st b, don't do anything bypass it.
for 2nd b, pop a
for 3rd b, bypass it.

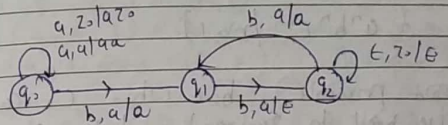
Date / /

Page No.:

Logic 1

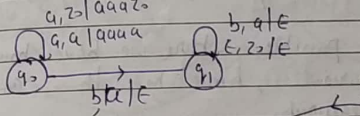


Logic 2

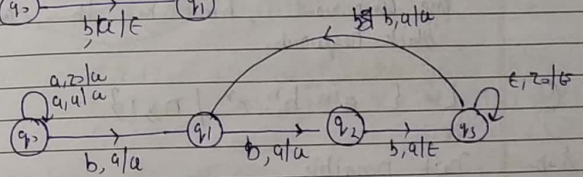


Q $L = \{ a^n b^{3n} \mid n \geq 1 \}$

Logic 1



Logic 2



Q $L = \{ a^n b^{n^2} \mid n \geq 1 \}$ (superscript power)

Sol: $L = \{ abb, a^2b^4, a^3b^9, \ldots \}$

PDA not possible.

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

Sol: $L = \{ ab, aabb, a^6b^6, a^{24}b^{24}, a^{120}b^{120}, \ldots \}$
not possible
if equal no of a's and b's concept is used then PDA will even accept a^2b^2 a^3b^3 a^4b^4 ie strings which are not in language

Scanned by CamScanner

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

Sol: PDA not possible

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

Sol: not possible, bcz when c's comes we need to remember both a's and b's. Hence 2 stacks required.

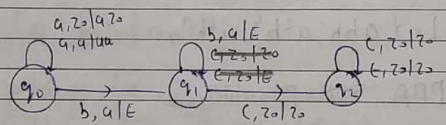
Q. $L = \{a^n b^{2n} c^n \mid n \geq 1\}$ (b should be 2 times of a & 2 times of c as well as # of a's and c's must be equal)

Sol: PDA not possible. Here we have more than one comparison, hence more than one one stack required.

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

Sol: not possible

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

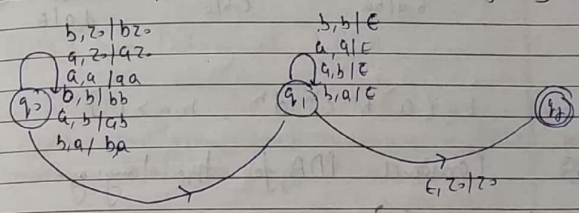


Q Identify language accepted by the following PDA

1) $\delta(q_0, a, Z_0) = (q_0, XZ_0)$
$\delta(q_0, a, X) = (q_0, XX)$
$\delta(q_0, b, X) = (q_1, \epsilon)$
$\delta(q_1, b, X) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, X) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, Z_0) = (q_1, \epsilon)$

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

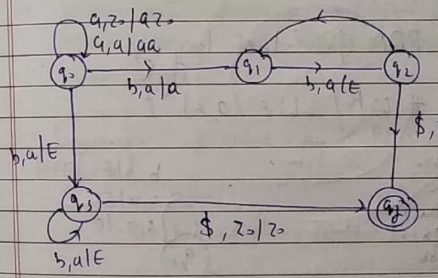
2) $L = \{x \mid x \in \{a,b\}^* \; n_a(x) \geq n_b(x)\}$



C,Z0|Z0
C,a|a
C,b|b

$L = \{(a+b)^n c (a+b)^n, n \geq 1\}$

3) $L = \{x c y, |x| = |y|$ $x, y \in \{a,b\}^*\}$



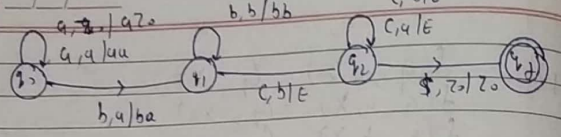
$L = \{a^n b^{2n}\} \cup \{a^n b^n\}, \; n \geq 1$

Scanned by CamScanner

Date / /

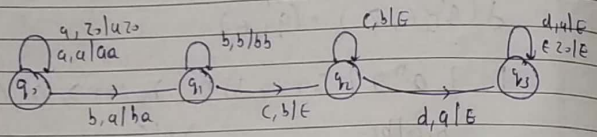
Page No.:

4)



$L = a^n b^m c^{n+m} \quad n, m \geq 1$

5)



$L = \{ a^n b^m c^m d^n \;,\; n, m \geq 1 \}$

6) Construct PDA for the language

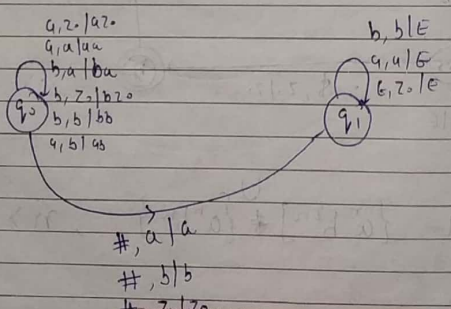
$L = \{ a^n b^m c^n d^m \mid n, m \geq 1 \}$

Sol: Not possible

Q Construct PDA for the language

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

Vol:

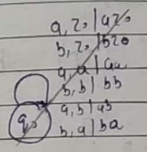


"To me, freedom and humanity is the right of everyone." –Sri Ramakrishna Paramahamsa

Date 10, 10, 18

Page No.:

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



next page

NOTE:

→ NPDA (Non Deterministic) PDA

In NPDA from the given state given i/p symbol and given stack symbol, finite no. of alternative transitions may exists.

→ Every DPDA is NPDA, but every NPDA need not be DPDA.

→ Expressive Power of NPDA is more than DPDA. Hence every NPDA can not be converted into DPDA.

→ Language accepted by PDA are classified into
i) DCFL's
ii) NDCFL's & NCFL's NCFL.

→ By default PDA means NPDA hence CFL's means NCFL.

"No real change in history has ever been acheived by discussions." –Subhash chandra bose

Scanned by CamScanner

Date __/__/____ Page No.: ____

- Every DCFL is CFL, but every CFL need not be DCFL.

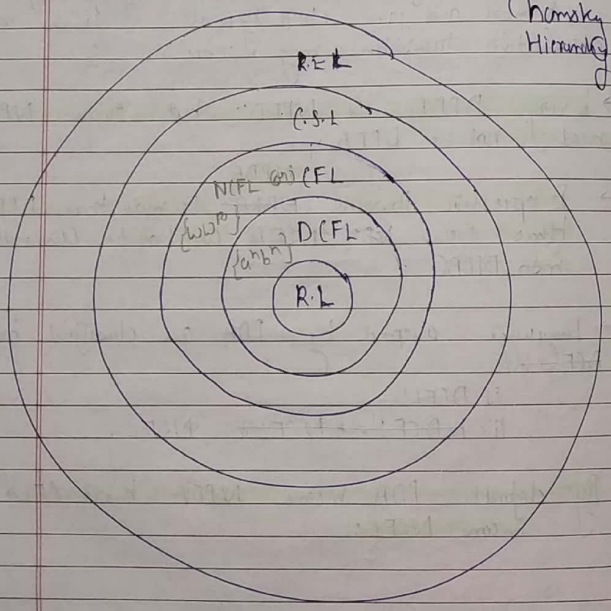
Imp following languages are CFL, but not DCFL.

i) $L_1 = \{ ww^R \mid w \in \{a,b\}^* \}$

ii) $L_2 = \{ a^n b^{2n} \} \cup \{ a^n b^{3n} \}$

iii) $L_3 = \{ a^n b^n c^m \} \cup \{ a^n b^m c^m \}$

Q



Date __/__/____ Page No.: ____

Q Which of the following languages is CFL but not DCFL?

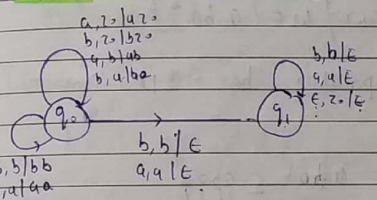
a) $L = \{ a^n b^n c d^m e^m / n,m \geq 0 \}$

b) $L = \{ a^n c b^n \} \cup \{ a^n d b^{2n} \}$

c) $L = \{ a^n b^n c^n / n \geq 1 \}$ ← not CFL

d) None

Q $L = \{ ww^R / w \in \{a,b\}^* \}$ (Vvv Important.) [IIT interview]



$\delta(q_0, \epsilon, Z_0) = (q_0, \epsilon)$
$\delta(q_0, a, Z_0) = \delta(q_0, aZ_0)$
$\delta(q_0, b, Z_0) = \delta(q_0, bZ_0)$
$\delta(q_0, a, b) = \delta(q_0, ab)$
$\delta(q_0, b, a) = \delta(q_0, ba)$

Note: w, w^R

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

$\delta(q_1, a, a) = \delta(q_1, \epsilon)$
$\delta(q_1, b, b) = \delta(q_1, \epsilon)$
$\delta(q_1, \epsilon, Z_0) = \delta(q_1, \epsilon)$

Scanned by CamScanner

Date __/__/____ Page No.: ____

NOTE: 1

i) All palindrom languages formed over more than one symbol are CFL but not regular.

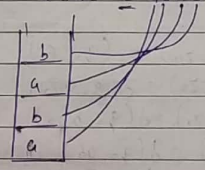
ii) ~~All~~ Palindrom languages formed over one symbol are CFL and regular

iii) Language recognization is difficult in NPDA compared to DPDA.

Q : $L = \{ wcw / w \in \{a,b\}^* \}$

Sol. PDA not possible, bcz LIFO not satisfied

Ex:- abab c abab



Q $L = \{ ww / w \in \{a,b\}^* \}$

Sol. Not possible.

Date __/__/____ Page No.: ____

NOTE:

i) Following two languages are non CFL

$L_1 = \{ ww / w \in \{a,b\}^* \}$

$L_2 = \{ wcw / w \in \{a,b\}^* \}$

but their complement is always CFL

$\Sigma^* - L_1$ } = CFL bcz CFGs exists for these

$\Sigma^* - L_2$ }

Q Construct PDA for the language ?

$L = \{ wcw^R \text{ id } w / w \in \{a,b\}^* \}$ (this w must be same as initial w)

Sol:- not Possible

Q $L = \{ ww^R w / w \in \{a,b\}^* \}$

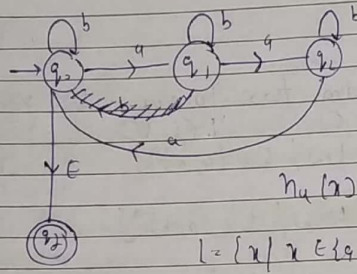
Sol: not possible

Q Identify language accepted by following PDA?

1.) $\delta(q_0, a, z_0) = (q_1, z_0)$ 7) $\delta(q_0, \epsilon, z_0) = (q_f, z_0)$
2.) $\delta(q_0, b, z_0) = (q_0, z_0)$
3.) $\delta(q_1, a, z_0) = (q_1, z_0)$ Here we don't use stack
4.) $\delta(q_1, b, z_0) = (q_1, z_0)$ We use only FA
5.) $\delta(q_2, a, z_0) = (q_0, z_0)$ Hence Lang must be R.L
6.) $\delta(q_2, b, z_0) = (q_2, z_0)$

Scanned by CamScanner

Date __/__/____ Page No.:



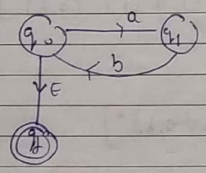
$n_a(x) \bmod 3 = 0$

$L = \{x \mid x \in \{a,b\}^* \; n_a(x) \bmod 3 = 0\}$

Q Construct Regular Expression equivalent to following PDA.

$\delta(q_0, a, Z_0) = (q_1, Z_0)$
$\delta(q_1, b, Z_0) = (q_0, Z_0)$
$\delta(q_0, \epsilon, Z_0) = (q_f, Z_0)$

Sol:



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

CFG to PDA Construction

CFG to PDA construction algorithm always results NPDA

Language generated by CFG's are non-deterministic CFL's. Hence for CFG we can construct NPDA always.

Date __/__/____ Page No.:

Any grammar is of the form $A \to \alpha$ $\alpha \in (V+T)^*$ then we can construct PDA as follows:-

1) $\delta(q_0, \epsilon, Z_0) = (q_1, SZ_0)$
2) $\delta(q_1, \epsilon, A) = (q_1, \alpha)$
3) $\delta(q_1, a, a) = (q_1, \epsilon)$
4) $\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$

Ex:- Consider CFG

$S \to AB$
$A \to ab/ba$
$B \to b$

Sol: P

$\delta(q_0, \epsilon, Z_0) = \delta(q_1, SZ_0)$

$\delta(q_1, \epsilon, S) = \delta(q_1, AB)$
$\delta(q_1, \epsilon, A) = \delta(q_1, ab)$ (or) $\delta(q_1, ba)$
$\delta(q_1, \epsilon, B) = \delta(q_1, b)$

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

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

This is the required P.D.A

Scanned by CamScanner

Date / /

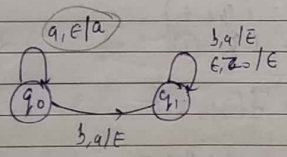
Page No.:

Q CFG

$S \rightarrow aA$
$A \rightarrow aABD \mid bB \mid a$
$B \rightarrow b$
$D \rightarrow d$

Sol: 1) $\delta(q_0, \epsilon, z_0) = \delta(q_1, Sz_0)$
2) $\delta(q_0, \epsilon, S) = \delta(q_1, aA)$
3) $\delta(q_1, \epsilon, A) = \delta(q_1, aABD)$ (or) $\delta(q_1, bB)$ (or) $\delta(q_1, a)$
6) $\delta(q_1, a, a) = \delta(q_1, \epsilon)$
7) $\delta(q_1, b, b) = \delta(q_1, \epsilon)$
8) $\delta(q_1, d, d) = \delta(q_1, \epsilon)$
4) $\delta(q_1, \epsilon, B) = \delta(q_1, b)$
5) $\delta(q_1, \epsilon, D) = \delta(q_1, d)$
9) $\delta(q_1, \epsilon, z_0) = \delta(q_f, z_0)$

Q Find the language accepted by following PDA



Sol: $\{a^n b^n, n \geq 0\}$

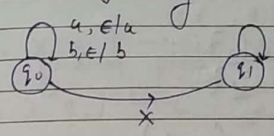
"To manifest the divinity is the right of everyone." –Sri Ramakrishna Paramahamsa

Date / /

Page No.:

Grade

Q Consider the following PDA.



odd length palindrome
$L = \{wXw^R / w \in \{a,b\}^*, X \in \{a,b\}\}$

Above PDA accepts set of all odd length plaindrom strings. Which of the following transitions is equal to X to accept odd length palindrom strings

a) $\delta(a, a/\epsilon)$, $\delta(b, b/b)$
b) $\delta(a, a/\epsilon)$, $\delta(b, b/\epsilon)$
c) $\delta(a, \epsilon/\epsilon)$, $\delta(b, \epsilon/\epsilon)$
d) none

PDA to CFG (Conversion) (Skip)

Q Construct an equivalent CFG for the following PDA

$\delta(q_0, 0, z_0) = (q_0, Xz_0)$
$\delta(q_0, 0, X) = (q_0, XX)$
$\delta(q_0, 1, X) = (q_1, \epsilon)$
$\delta(q_1, 1, X) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, X) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

"No real change in history has ever been achieved by discussions." –Subhash chandra bose

Scanned by CamScanner

Start Symbol

$S \to \delta[q_0, Z_0, q_2]$ } state possibility
$S \to \delta[q_0, Z_0, q_1]$

$\delta(q_0, 1, X) = (q_1, \epsilon)$
$[q_0, X, q_1] \to 1$

pop op

$\delta(q_1, 1, X) = (q_1, \epsilon)$
$[q_1, X, q_1] \to 1$

$\delta(q_1, \epsilon, X) = (q_1, \epsilon)$
~~$[q_1, \epsilon, q_1]$~~ $[q_1, X, q_1] \to \epsilon$

$\delta(q_1, \epsilon, Z_0) = (q_1, \epsilon)$
$[q_1, Z_0, q_1] \to \epsilon$

$\delta(q_0, 0, Z_0) = (q_0, XZ_0)$ Push Opn

$[q_0, Z_0, q_2] \to 0[q_0, X, q_0][q_0, Z_0, q_2]$ Same

$[q_0, Z_0, q_2] \to 0[q_0, X, q_1][q_1, Z_0, q_2]$

$[q_0, Z_0, q_1] \to 0[q_0, X, q_0][q_0 Z_0 q_1]$

$[q_0, Z_0, q_1] \to 0[q_0, X, q_1][q_1, Z_0, q_1]$

$\delta(q_0, 0, X) = (q_0, XX)$

$[q_0, X, q_0] \to 0[q_0, X, q_0][q_0 X q_0]$

$[q_0, X, q_0] \to 0[q_0, X, q_1][q_1 X q_0]$

$[q_0, X, q_1] \to 0[q_0, X, q_0][q_0 X q_1]$

$[q_0, X, q_1] \to 0[q_0, X, q_1][q_1, X, q_1]$

$[q_0 Z_0 q_0] = A$ $[q_1 Z_0 q_0] = C$

$[q_0 Z_0 q_1] = B$ $[q_1, Z_0, q_1] = D$

$[q_0 X q_0] = E$ $[q_0, X q_1] = F$

$[q_1 X q_0] = G$ $[q_1 X q_1] = H$

$S \to A | B$
$F \to 1$
$H \to 1 | \epsilon$
$D \to \epsilon$
$A \to 0EA | 0FC$
$B \to 0EB | 0FD$
$E \to 0EE | 0FG$
$F \to 0EF | 0FH$

from the obtained grammar remove useless symbols.

$S \to B$ $F \to 0FH$
$F \to 1$
$H \to 1 | \epsilon$
$D \to \epsilon$
$B \to 0FB$

Scanned by CamScanner

Date / /

Page No.:

Q

$(q_0, 1, Z_0) = (q_0, XZ_0)$ push
$(q_0, 1, X) = (q_0, XX)$ push
$(q_0, 0, X) = (q_1, X)$
$(q_1, 1, X) = (q_1, \epsilon)$ pop
$(q_1, 0, Z_0) = (q_0, Z_0)$
$(q_0, \epsilon, Z_0) = (q_0, \epsilon)$ pop

$S \rightarrow [q_0, Z_0, q_0]$
$S \rightarrow [q_0, Z_0, q_1]$

$\delta(q_1, 1, X) = (q_1, \epsilon)$

push

$[q_1, X, q_1] \rightarrow 1$

$\delta(q_0, \epsilon, Z_0) = (q_0, \epsilon)$

$(q_0, Z_0, q_0) \rightarrow \epsilon$

$\delta(q_0, 0, X) = (q_1, X)$

by push

$[q_0, X, q_0] = 0\,[q_1, X, q_0]$

same as push

$[q_0, X, q_1] = 0\,[q_1, X, q_1]$

$\delta(q_1, 0, Z_0) = (q_0, Z_0)$

$[q_1, Z_0, q_0] = [q_0, Z_0, q_0]$

$[q_1, Z_0, q_1] = [q_0, Z_0, q_1]$

Date / /

Page No.:

$\delta(q_0, 1, Z_0) = (q_0, X, Z_0)$

$[q_0, Z_0, q_0] \rightarrow 1\,[q_0, X, q_0]\,[q_0, Z_0, q_0]$
$[q_0, Z_0, q_0] \rightarrow 1\,[q_0, X, q_1]\,[q_1, Z_0, q_0]$
$[q_0, Z_0, q_1] \rightarrow 1\,[q_0, X, q_0]\,[q_0, Z_0, q_1]$
$[q_0, Z_0, q_1] \rightarrow 1\,[q_0, X, q_1]\,[q_1, Z_0, q_1]$

$\delta(q_0, 1, X) = (q_0, XX)$

$[q_0, X, q_0] \rightarrow 1\,[q_0, X, q_0]\,[q_0, X, q_0]$
$[q_0, X, q_0] \rightarrow 1\,[q_0, X, q_1]\,[q_1, X, q_0]$
$[q_0, X, q_1] \rightarrow 1\,[q_0, X, q_0]\,[q_0, X, q_1]$
$[q_0, X, q_1] \rightarrow 1\,[q_0, X, q_1]\,[q_1, X, q_1]$

$[q_0, Z_0, q_0] = A$
$[q_0, Z_0, q_1] = B$
$[q_0, X, q_0] = E$
$[q_0, X, q_1] = F$
$[q_1, X, q_0] = G$
$[q_1, X, q_1] = H$
$[q_1, Z_0, q_0] = C$
$[q_1, Z_0, q_1] = D$

$S \rightarrow A/B$
$H \rightarrow 1$
$A \rightarrow \epsilon/1FC/1EA$
$E \rightarrow 0G/1FG/1EE$
$F \rightarrow 0H/1EF/1FH$
$C \rightarrow 0A$
$D \rightarrow 0B$
$B \rightarrow 1EB/1FD$

$S \rightarrow A$
$H \rightarrow 1$
$A \rightarrow \epsilon/1FC$
$F \rightarrow 0H/1FH$
$C \rightarrow 0A$

Scanned by CamScanner

11/10/18

Context Free Language

(V.V Imp)

Q2 Which of the following language is

(1) CFL & Regular

(2) CFL but not regular

(3) Non CFL.

1) $L = \{ a^n b^n c^n \mid 1 \le n \le 1000 \}$ (1)

2) $L = \{ a^n b^n c^m \mid n \neq m \}$ (3)

3) $L = \{ a^n b^n c^m d^m \mid n, m \geq 1 \}$ (2)

4) $L = \{ a^n b^n c^m d^m \mid n \neq m \}$ (3)

5) $L = \{ a^n b^{m+n} c^m \mid n, m \geq 1 \}$ (2)

6) $L = \{ a^n b^{m+n} c^{m+n} \mid n, m \geq 1 \}$ (3) multiple comparison n_a must be $< n_b$ & n_c.

7) $L = \{ a^{n+m} b^{n+m} c^m \mid n, m \geq 1 \}$ (3)

8) $L = \{ a^n b^m \mid n - m = 4 \}$ (2)

9) $L = \{ a^n b^m \mid \frac{n}{m} = 5 \}$ (2)

10) $L = \{ a^n b^m \mid n = 2m+1 \}$ (2)

11) $L = \{ a^n b^m \mid n = m^2 \}$ (3)

12) $L = \{ a^n b^m \mid n \neq m \}$ (2)

Scanned by CamScanner

13) $L = \{ a^i b^j c^k \mid i+k = j \}$ ②

14) $L = \{ a^i b^j c^k \mid j = \max\{i,k\} \}$ ①

15) $L = \{ a^i b^j c^k \mid i > j > k \}$ ③

16) $L = \{ a^i b^j c^k \mid i \geq j \text{ (or) } j < k \}$ ② NPDA

17) $L = \{ a^i b^j c^k \mid i = j^2 + k^2 \}$ ③

18) $L = \{ a^n b^n c d^m e^n \mid n, m \geq 1 \}$ ③ more than 1 stack required

19) $L = \{ a^n b^m c^m d^n \mid n, m \geq 1 \}$ ②

20) $L = \{ a^n b^m c^n d^m \mid n, m \geq 1 \}$ ③ more than 1 stack required

21) $L = \{ a^n b^n c^m d^m \mid n \neq m \}$ ③

22) $L = \{ a^n b^{2n} c^{3m} \mid n, m \geq 1 \}$ ②

23) $L = \{ a^i b^j c^k d^l \mid i = k \text{ (or) } j = l \}$ ② NPDA

24) $L = \{ a^i b^j c^k d^l \mid i = 2j \text{ and } k = 3l \}$ ②

25) $L = \{ a^i b^j c^k d^l \mid i \neq k \text{ and } j \neq l \}$ ③

26) $L = \{ a^i b^j c^k d^l \mid i+j = k+l \}$ ②

27) $L = \{ a^i b^j c^k d^l \mid i^2 + j^2 = k^2 + l^2 \}$ ③

28) $L = \{ a^p \mid p \text{ is a prime no.} \}$ ③

29) $L = \{ a^k \mid k \text{ is even number} \}$ ①

30) $L = \{ 1, 2, 4, 8, \ldots 2^n \ldots \}$ ③
these numbers in binary

31) $L = \{ ww \mid w \in \{a^*\} \}$ ①

32) $L = \{ ww \mid w \in \{a,b\}^* \}$ ③

33) $L = \{ wxw \mid w, x \in \{a,b\}^* \}$ ① RE = $(a+b)^*$

34) $L = \{ \Sigma^* - \{ ww \mid w \in \{a,b\}^* \} \}$ ②

35) $L = \{ ww^R ww^R \mid w \in \{a,b\}^* \}$ ③ must be same

36) $L = \{ ww^R ww^R ww^R \mid w \in \{a,b\}^* \}$ ①

37) $L = \{ x \mid x \in \{a,b\}^* \; n_a(x) = 3 n_b(x) \}$ ②

38) $L = \{ x \mid x \in \{a,b\}^* \; n_a(x) \neq n_b(x) \}$ ②

39) $L = \{ x \mid x \in \{a,b,c\}^* \; n_a(x) = n_b(x) = n_c(x) \}$ ③

40) $L = \{ x \mid x \in \{a,b,c\}^* \; n_a(x) + n_b(x) = n_c(x) \}$ ②

41) $L = \{ x \mid x \in \{a,b,c\}^* \; n_a^2(x) + n_b^2(x) = n_c(x) \}$ ③

42) $L = \{ x \mid x \in \{a,b\}^* \; n_a(x) \geq n_b^2(x) \}$ ③

43) $L = \{ a^{2n} b^n c^{n+m} \mid n, m \geq 1 \}$ ③

44) $L = \{ a^{2n} b^{3n} c^{5n} \mid n \geq 1 \}$ ③

45) $L = \{ a^n b^m c^k \mid n \neq m \text{ (or) } m \neq k \}$ ② NPDA

46) set of all odd length palindrom strings of finite language. ②

Scanned by CamScanner

47) Set of all even length palindrom strings of Braj (2)

48) Set of all balanced parenthesis. (2)

49) Set Equal no. of open and closing parenthesis (2)

50) Set of all lexical errors detected by compiler. (1)
CFL & Regular

51) Set of all syntax errors detected by compiler (CFL (2)

52) Set of all ~~system~~ semantics errors detected by compiler. CSL (3)

53) CFL Detection

1) Any finite language is regular and CFL

2) There is no difference b/w CFL & regular if the language is formed over one symbol.

3) Hence over one symbol languages strings are in A.P, CFL otherwise non CFL

4) All palindrom languages are CFL.

5) Palindrom languages formed over more than one symbol are CFL but not regular and formed over one symbol are CFL & regular

6) Any infinite language requires more than one stack then it is non-CFL (more than one comparision exists)

7) Any infinite language not satisfy stack property then it is non-CFL (LIFO order)

8) Union of 2 CFL's is also CFL.

Grade Q Which of the following language is CFL?

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

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

c) $L = \{ a^m b^n c^p d^q / m = n = p = q \}$

d) $L = \{ a^n b^n c^p d^q / n = m \text{ and } p = q \}$

Imp

Closure Properties of CFL and DCFL

1) Subset Operation

Subset of CFL is may or may not be CFL
Hence CFL is not closed under subset opn

Ex: $\{a^n b^n c^n\} \subseteq \{a^n b^n c^m / n, m \geq 1\}$ may not
not CFL

$\{a^n b^n\} \subseteq \{a^n b^n c^m\}$ may
CFL

Scanned by CamScanner

DCFL

Subset of DCFL may or may not be DCFL
Hence DCFL is not closed under subset oph.

Ex: $\{a^n b^n c^n\} \subseteq \{a^n b^m c^m \mid n, m \geq 1\}$

not CFL — DCFL

2) Union Operation

(CFG's)

$L_1 = CFL = \{a^n b^n\}$ $\quad S_1 \rightarrow aS_1 b \mid ab$

$L_2 = CFL = \{c^n d^m\}$ $\quad S_2 \rightarrow cS_2 d \mid cd$

$L = L_1 \cup L_2 = \{a^n b^n\} \cup \{c^n d^n\} \rightarrow S \rightarrow S_1 \mid S_2$

Union of 2 CFL's is always CFL. Hence CFL's are closed under union oph. (CFG for $L_1 \cup L_2$ is $S \rightarrow S_1 \mid S_2$)

3) DCFL: Union of 2 DCFL may or may not be DCFL. Hence DCFL's are not closed under union oph.

Ex. $L = \{a^n b^{2n}\} \cup \{a^n b^{3n}\}$

✓ DCFL — DCFL

union DCFL

3) Intersection Operation

Intersection of 2 CFL's may or may not CFL. Hence CFL's are not closed under intersection oph.

Ex $L = \{a^n b^m c^m\} \cap \{a^n b^n c^m\} = \{a^n b^n c^n\}$

CFL — CFL — not CFL

DCFL

Intersection of 2 DCFL's may or may not DCFL. Hence DCFL's are not closed under intersection oph.

Ex $\{a^n b^n c^m\} \cap \{a^n b^m c^m\} = \{a^n b^n c^n\}$

DCFL — DCFL — not CFL

4) Intersection with Regular language

CFL $\cap$ RL is always CFL. but may (or) may not be Regular.

Hence CFL's are closed under intersection with Regular lang. oph. CFL $\cap$ RL = CFL

Ex: $\{a^n b^n \mid n \geq 1\} \cap (a+b)^* = \{a^n b^n \mid n \geq 1\}$

(CFL) (RL) (CFL)

$\{a^n b^n \mid n \geq 1\} \cap (a^* b^*) = \{a^n b^n \mid n \geq 1\}$

(CFL)

$\{a^n b^n \mid n \geq 1\} \cap \{ab\} = \{ab\}$

CFL & RL

Scanned by CamScanner

Generalized

$$X \cap R.L = X$$

$$X \in \{ CFL, \overset{DCFL}{\sim} CSL, REL \}$$

Hence this oph is closed under all formal languages

DCFL : Hence DCFL $\cap$ R.L is always DCFL

$\therefore$ DCFL also closed under this oph.

Gate

Q Consider the following 2 languages L_1 & L_2

$L_1 = \{ a^n b^n c a^m \mid n, m \geq 0 \}$

$L_2 = \{ a^i b^j c^k \mid i, j, k \geq 1 \}$

$L_1 \cap L_2 = ?$

a) CFL & regular

b) CFL but not regular

c) Non CFL

d) R.L but not CFL

Concatination oph

Concatination of 2 CFL's is always CFL bcz we can construct CFG for $L_1 \cdot L_2$ as

$S \rightarrow S_1 \cdot S_2$

$L_1 = \{ a^n b^n \}$ $S_1 \rightarrow aS_1b/ab$

$L_2 = \{ c^m d^m \}$ $S_2 \rightarrow cS_2d/cd$

$L = L_1 \cdot L_2 = \{a^n b^n\} \cdot \{c^m d^m\}$ $S \rightarrow S_1 \cdot S_2$

Hence CFL's are closed under concatination oph.

DCFL : Concatination of 2 DCFL's is may or may not be DCFL Hence DCFL's are not closed under concatin oph.

Ex:- $\{ X \cdot W C W^R \mid W, X \in \{a,b\}^+ \}$

↑ not DCFL

Complimentation oph

Compliment of CFL may or may not be CFL. Hence CFL's are not closed under compliment oph.

Scanned by CamScanner

$\Sigma^* - \{ww \mid w \in \{a,b\}^*\} \rightarrow$ CFL

Complement $= \Sigma^* - (\Sigma^* - \{ww \mid w \in \{a,b\}^*\})$

$= \{ww \mid w \in \{a,b\}^*\} \rightarrow$ not CFL

NOTE: Complement of CFL is recursive language

DCFL : Complement of DCFL is always DCFL because we can complement DPDA by interchanging final & non-final state. (but we can not complement NPDA)

DCFL = DPDA

$\Sigma^* - DCFL = DPDA^C$

Difference Operation

$L_1 \Rightarrow$ CFL $\quad L_2 \Rightarrow$ CFL

Intersection Difference of 2 CFL may or may not be CFL because $L_1 - L_2$ is $L_1 \cap L_2^C$

$L_1 - L_2 = L_1 \cap L_2^C$

CFL's are not closed under $\cap$ and complement Hence also not closed under difference.

$L_1 \Rightarrow$ DCFL $\quad L_2 \Rightarrow$ DCFL

$L_1 - L_2 = L_1 \cap L_2^C$

Difference of 2 DCFL may or may not be DCFL, Hence DCFL's are not closed under difference opn. ($\cap$ not closed Hence difference not closed)

Difference with Regular language

$CFL - R.L = CFL \cap R.L^C$

$\Rightarrow CFL \cap Reg \Rightarrow$ always CFL

CFL difference with Regular is always CFL but may or may not be regular.

Hence CFL's are closed under difference with R.L opn

Generalized $X \in \{CFL, DCFL, CSL, REL\}$

$X - R.L = X$

Hence this opn is closed under all formal languages.

DCFL

DCFL $-$ Reg is always DCFL

Hence DCFL's are also closed under this opn.

Scanned by CamScanner

Q. The following language is

$L = \{a^n b^n / n \geq 1 \& n \neq 13\}$

a) DCFL

b) CFL but not DCFL

c) non CFL

d) Recursive but not CFL.

(except) $\{a^n b^n / n \geq 1\} - \{a^n b^n / n = 13\}$

DCFL — Reg = DCFL

Reversal op^n

Reverse of a CFL is always CFL. bcz we can construct CFG for L^R by reversing R.H.S part of given CFG.

Hence CFL's are closed under Reversal op^n

$L = \{a^n b^n / n \geq 1\}$ $S \to aSb / ab$

$L^R = \{b^n a^n / n \geq 1\}$ $S \to bSa / ba$

DCFL

Reverse of a DCFL is may or may not DCFL

Hence DCFL is not closed under DCFL op^n.

$L = \{ w c w^R x / w, x \in \{a, b\}^+ \}$ ← DCFL

$L^R = \{ x w^R c w / w, x \in \{a, b\}^+ \}$ ← not DCFL

Klein closure op^n & Positive Klein closure

Klein closure of a CFL is always CFL.

Positive Klein closure of a CFL is always CFL.

Hence CFL's are closed under * and +.

DCFL

Klein closure of DCFL may or may not be DCFL

Positive Klein closure may or may not be DCFL

Hence DCFL's are not closed under * and +

Ex $L \Rightarrow$ CFL S_1

$L^* = S \to S_1 S / \epsilon$

DCFL

$L^+ = S \to S_1 S / S_1$

Scanned by CamScanner

	CFL	DCFL
1) ~~Intersection~~	✓	×
2) Union	✓	×
3) Concatination	✓	✓
4) $(\Sigma^* - L)$	×	×
5) L^*	✓	×
6) L^+	✓	×
7) $L \cap Reg$	✓	✓
8) $L_1 - L_2$	×	×
9) $L - Reg$	✓	✓
10) L^R	✓	×
11) Substitution	✓	×
12) Homomorphism	✓	×
13) Inverse Homo-morphism	✓	✓
14) L_1 / L_2	×	×
15) L / Reg	✓	✓
16) Init	✓	✓
17) $L \cup Reg$	✓	✓
1) Intersection	×	×

Q Let P be CFL & Q is Regular language then which of the following is always regular

a) $P \cap Q$

b) $P - Q$

c) $\Sigma^* - P$

~~d)~~ $\Sigma^* - Q$

Q Consider the following languages

$L_1 = \{ a^n b^n c^m \mid n, m \geq 1 \}$

$L_2 = \{ a^n b^m c^m \mid n, m \geq 1 \}$

Which of the following is false?

a) $L_1 \cup L_2$ is CFL

b) $L_1 . L_2$ is CFL

~~c)~~ $L_1 \cap L_2$ is CFL

d) none of these

Q L_1 & L_2 are DCFL

L_3 & L_4 are CFL

& L_5 is Regular

Scanned by CamScanner

Which of the following is false

a) $L_1^C - L_5$ is DCFL

b) $(L_3 \cup L_4)^C$ is Recursive

c) $(L_3^R \cup L_4) \cap L_5^*$ is CFL

d) none

Q Let L_1 & L_2 are CFL.
Let L_3 & L_4 are RL
Which of the following is always CFL?

1) $(L_1 \cup L_2)^* \cap (L_3 \cap L_4)^R$

2) $(L_1 \cup L_3)^* \cap (L_2 \cap L_4)^R$ ✗

3) $(L_1 \cdot L_2) \cap (L_3 \cdot L_4)$

4) $(L_1 \cdot L_3) \div (L_2 \cdot L_4)$ ✗
$((CFL \cdot Reg) \Rightarrow$ CFL bcz every R.L is CFL by chomsky Hierarchy

5) $(L_1 - L_3) \cdot (L_2 - L_4)$

6) $(L_1 - L_3) - (L_2 - L_4)$ ✗

7) $(L_1 / L_3) \cup (L_2 / L_4)$

8) $(L_1 / L_4) / (L_2 / L_3)$ ✗

9) $(L_1 \cdot L_3) / (L_2 \cdot L_4)$ ✗

10) $(L_1 L_2) / (L_3 \cdot L_4)$

$(L_1 L_2) / L_3 \cdot L_4$

Q L_1 is Regular, L_2 is DCFL, L_3 is CFL
Which of the following is false.

a) $L_1 \cup L_2 \cup L_3$ is CFL T

b) $L_1 \cap L_2 \cdot \cap L_3$ is CFL T

c) $L_1 \cap$

b) $L_1 \cdot L_2 \cdot L_3$ is CFL T

c) $L_1 \cap L_2 \cap L_3$ is CFL F
$\{a^n b^n c^m\} \cap \{a^n b^m c^m\}$

d) none

Decision Properties of CFL

Following problem decidable under CFL

1) Emptiness problem
2) Finiteness problem
3) Membership problem

Scanned by CamScanner

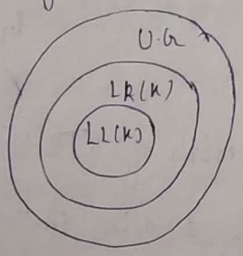
NOTE:
Equivalence problem of CFL is undecidable problem

Grammars for DCFL

1) Parser is a DPDA. Hence grammars suitable for parser generates DCFL's.

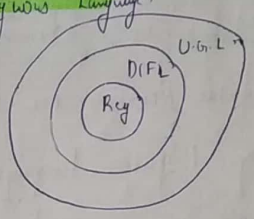
2) LL(k) grammar and LR(k) grammar are grammars suitable for Top down parser and bottom-up parser respectively.

3) Every LL(k) grammar is LR(k) grammar, but Every LR(k) grammar need not be LL(k)



4) Every LL(k) grammar, LR(k) grammar is Unambiguous grammar Hence every DCFL is also unambiguous.

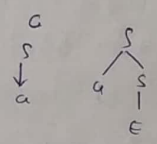
5) Every R.L is DCFL, Hence every R.L is also unambiguous Language.



U.G.L
Unambiguous grammar

7) Regular grammar can be ambiguous.

$S \to aS \mid a \mid \in$



8) A language is Unambiguous because even though multiple grammars exists for that language but at least one unambiguous grammar exists.

9) Regular language and DCFL's are not inherently ambiguous language.

10) For every DCFL LR(k) grammar exists but LL(k) grammar may (or) may not exists.

11) For every R.L also LR(k) grammar exists but LL(k) grammar may (or) may not exists

Scanned by CamScanner

Q Which of the following is true?

a) All unambiguous grammars are regular grammars.

b) All regular grammars are unambiguous grammars

c) For every regular language unambiguous grammar should exists.

d) None

Decision properties of DCFL

following problems are decidable under DCFL

1) Membership problem (CYK Algorithm)
2) Finiteness problem (CNF graph)
3) Emptiness problem (Useless Variable algorithm)

NOTE: Equivalence problem of DCFL is decidable [parsing method]

NOTE: No. of languages accepted by DPDA with final state acceptance method is more than DPDA with Empty Stack Method

2) If a language not having prefix property then it is accepted by final state method only but not by empty stack method of DPDA

Prefix Property

a language is said to have prefix property if proper prefix of any string is not present in the language.

The following language having prefix property

$L_1 = \{a^n b^n \mid n \geq 1\}$

{ab, aabb, aaabbb.... }

ϵ ϵ ϵ
a a a
aa aa
aab aaab

The following language not having prefix property

$L_2 = \{a^n b^n \mid n \geq 0\}$

{ϵ, ab, aabb, ... }

ϵ
a

Scanned by CamScanner

→ language L_2 is accepted by DPDA final state method only but not by empty stack method (not having prefix property).

→ Language L_1 accepted by empty stack and final state method (having prefix property)

Equivalence Classes

1) Number of equivalence classes corresponding to given R.L is equal to total no of states of minimal DFA possible for the language.

Q. How many total no. of equivalence classes possible for the following R.L

① $(a+b)^* ab = 3$

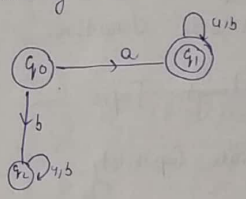
② $(a+b)^* a(a+b)(a+b) = 8$

③ $(a+b)^* aba(a+b)^* = 4$

→ Equivalence class of a state q_i in any DFA is set of all strings by which we can reach each state q_i starting from initial state.

→ Union of equivalence classes of all states of the a DFA is equal to complete language.

Q Calculate Equivalence class of every state for the following DFA.



Eq. class $(q_0) = \epsilon$

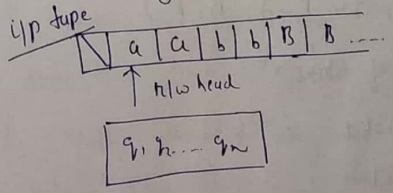
Eq. class $(q_1) = a(a+b)^*$

Eq. class $(q_2) = b(a+b)^*$

Eq. class $(q_0) \cup$ Eq. class $(q_1) \cup$ Eq. class $(q_2) = \Sigma^*$

Turing Machine

Turing machine is a mathematical model used to study behaviour of computer



Scanned by CamScanner

→ following are capabilities of Turing m/c

1) Turn around :-
   TM capable to move left as well as right side direction.

2) ~~Infinite length Tape~~

2) Read / Write capability :-
   TM reads symbols from i/p tape, replaces it by same symbol or some other symbol.

3) i/p tape of TM is One side closed. Otherside is infinite in direction.

→ TM can be represented by using Transition diagram (or) by Transition table.

→ Formal definition of TM is

TM = (Q, Σ, q_0, F, δ, B, Γ)

Q = finite no. of States

Σ = input alphabet Σ = {a, b}

q_0 = initial state

F = set of final states

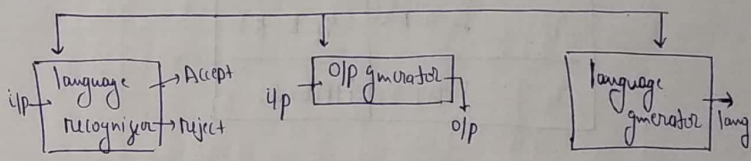
B = blank symbol

Γ = tape alphabet {a, b, X, Y, B}

δ : Transition function δ(q, X) = δ(q, Y, R)

$Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

Types of TM



Language Recognizer

→ By reading the i/p string TM may or may not halts.

→ By reading the i/p string TM halts in final state then given string is accepted.

→ By reading the i/p string TM halts in Non-final state then given string is rejected.

→ By reading the i/p string if TM enters into infinite loop then we can not conclude. God whether the i/p is accepted (or) not (God knows)

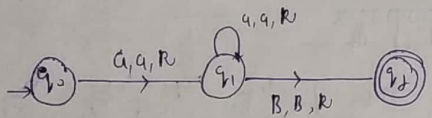
Scanned by CamScanner

Q Construct TM for the language

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

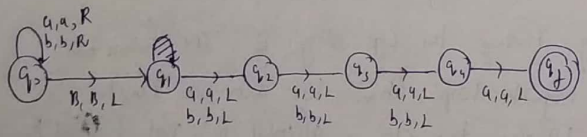
Sol: $Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

	a	B
q_0	(q_1, a, R)	
q_1	(q_1, a, R)	(q_f, B, R)
q_f	HALT	HALT



Q Construct TM that accepts all strings of a's and b's where ~~5th~~ 4th i/p symbol is "a" while reading the string from Right Hand side.

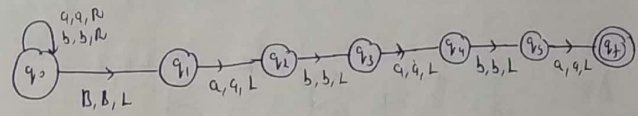
Sol:- $(a+b)^* a (a+b)(a+b)(a+b)$



Q Construct the TM for the language

$L = (a+b)^* ababa$

Sol:-

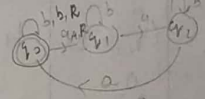


Q Language accepted by following TM

①

	a	b	B
→ q_0	(q_1, a, R)	(q_1, b, R)	
q_1	(q_2, a, R)	(q_2, b, R)	
q_2		(q_3, b, R)	
q_3	(q_3, a, R)	(q_3, b, R)	(q_f, B, R)
q_f	HALT		

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



②

	a	b	B
→ q_0	(q_1, a, R)	(q_0, b, R)	(q_f, B, R)
q_1	(q_2, a, R)	(q_1, b, R)	
q_2	(q_0, a, R)	(q_2, b, R)	
q_f	HALT		

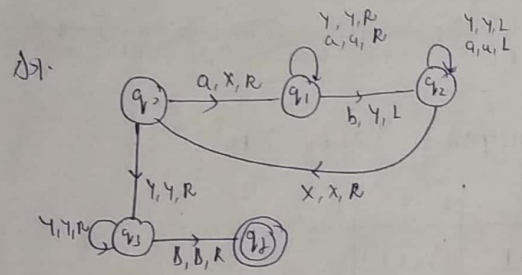
$n_a(1) \bmod 3 = 0$

Scanned by CamScanner

Q Construct TM for the language

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

Sol:



aaa bbb
Xaa Ybb
XXa YYb

	a	b	X	Y	B
q_0	(q_1, X, R)	H	H	(q_2, Y, R)	H
q_1	(q_1, a, R)	(q_2, Y, L)	H	(q_1, Y, R)	H
q_2	(q_2, a, L)	H	(q_0, X, R)	(q_2, Y, L)	H
q_3	H	H	H	(q_3, Y, R)	(q_f, B, R)

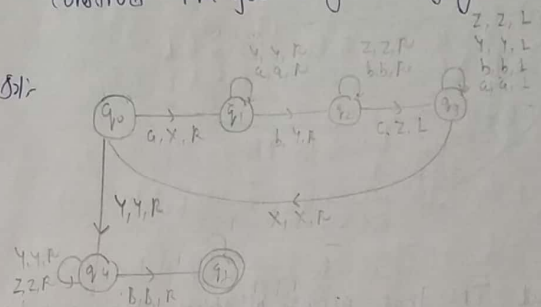
HALT

12/10/18

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

Construct TM for the given language.

Sol:-



Q What are the halting states for the following i/p strings corresponding to above TM?

① abcabc ② aabc ③ ababc

Sol: ① XYZabc → q_3

② XXYZ → q_1

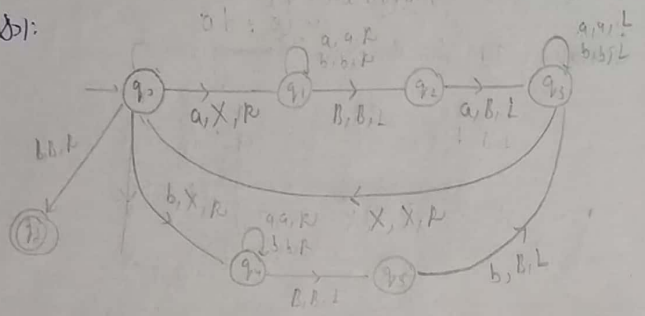
③ XbYbZ → q_4

Q Construct the TM that accepts all strings of a's and b's where each string is even length palindrom.

Scanned by CamScanner

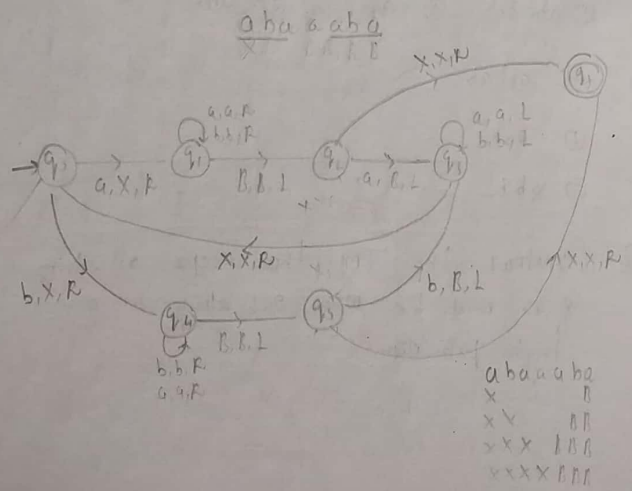
$L = \{ WW^R \mid W \in (a,b)^* \}$

Sol:



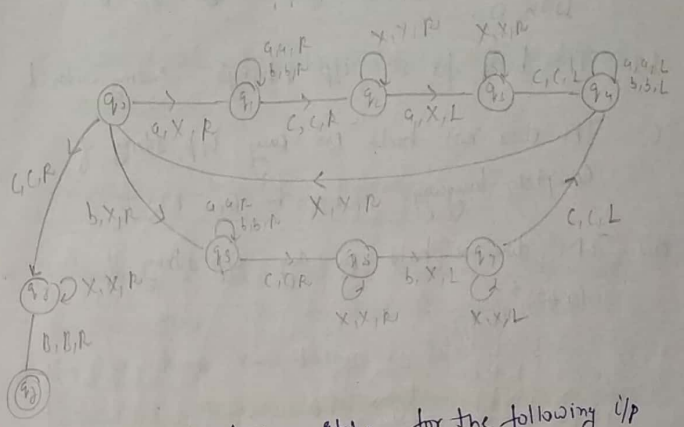
Q Construct TM that accepts all strings of a's and b's where each string is odd length palindrom.

Sol:- $L = \{ W(a+b)W^R \mid W \in (a,b)^* \}$



Q Construct the TM for the language

$$L = \{ WcW / W \in (a,b)^* \}$$



Q What are the halting states for the following i/p strings corresponding to above TM

① abc abc ② aabcaab ③ abc ba

Sol: ① abc abc q_8

② aabcaab q_2

③ abc ba q_2

Grade

Q Consider the following TM

	0	1	B
q_0	$(q_1, 1, R)$	$(q_1, 1, R)$	H
q_1	$(q_1, 1, R)$	$(q_0, 1, L)$	(q_2, B, L)

Scanned by CamScanner

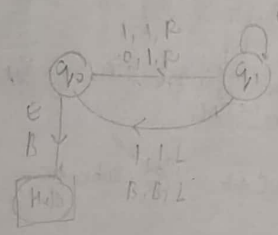
Q. Which of the following is True?

a) TM halts for all i/p strings ending with 0.

b) TM halts for all i/p strings ending with 1

c) TM does not halt on any i/p string of complete language → halts on ε means ε

d) TM does not halt on any i/p string of $(0+1)^+$



Recursive language

A language is said to be recursive if there exists a TM for that language that halts always for all i/p string.

ie for valid strings TM halts in final state, for invalid string TM halts in non final state.

→ Halting TM is exactly same as algorithm. Hence Recursive languages are also called as Turing decidable languages.

→ If a language is recursive corresponding problem is decidable

Recursive Enumerable languages (R.E.L)

→ A language is said to be Recursive Enumerable if there exists TM for that language that halts for i/p strings and may not halt for some i/p string ie for valid strings TM halts in final state. For invalid strings TM may halt in non-final state, or may enter into infinite loop.

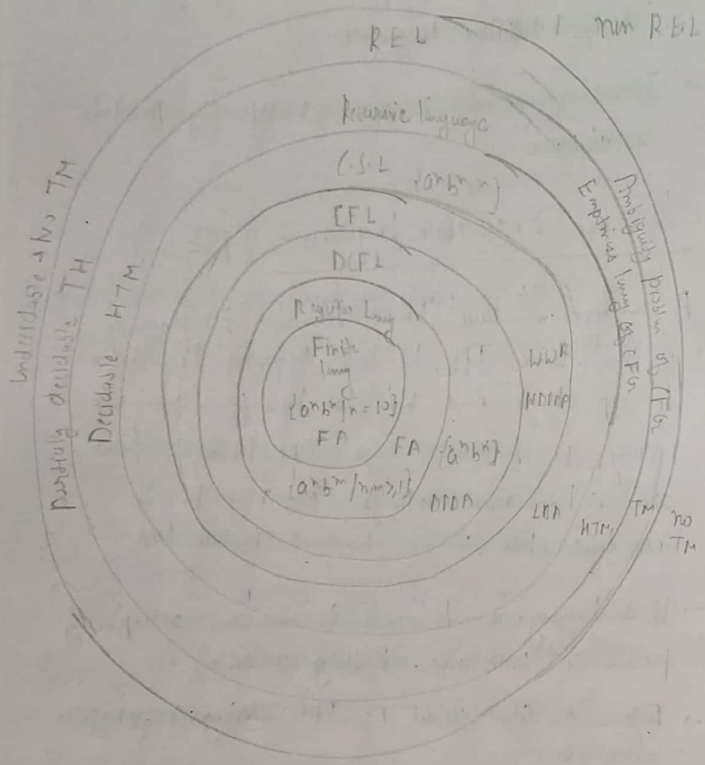
→ If a language is Recursive Enumerable, corresponding problem is undecidable. (Partially decidable)

→ R.E.L's also called as Turing recognizable languages.

→ By default TM means may (or) may not halting m/c. Hence by default language of TM are R.E.L's.

Scanned by CamScanner

→ Every Recursive language is R.E.L. but every R.E.L need not be Recursive.



Countably Infinite Set

A set is said to be countable if there exists 1-1 correspondance with natural set to the given set.

Ex:- Set of natural numbers

$N = \{1, 2, 3, \ldots\}$

Set of Complete language

$\Sigma^* = \{\epsilon, 0, 1, 00, \ldots 11, \ldots\}$

Total population of the world

Uncountable Set

A set is said to be ~~to be~~ uncountable if there is no one to one correspondence with natural set. to the given set. (Size of the set ~~more~~ is more than natural set)

Ex: Set of Real No's.

Power set of natural set 2^N

Power set of Complete Set 2^{Σ^*}

Set of all languages possible over i/p alphabet Σ

Scanned by CamScanner

→ If the set is uncountable no TM possible for that set, so Hence all uncountable sets are non-R.E.L

→ If the language is non-REL then it is undecidable.

→ Total no. of undecidable problems are uncountable

→ Total no of decidable problems are countable

→ Total no of languages for which we can construct TM are countable

→ Total no of problems, not solvable by computers are uncountable

Q Which of the following is countable?

a) Set of all languages possible over some alphabet Σ. uncountable

b) No. of points in a line. uncountable

c) Power set of complete language uncountable

d) No of Hairs on person head countable

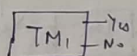
13/10/18

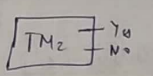
Closure Properties of Rec. lang. & Recursive Enumerable languages.

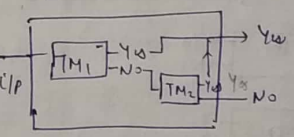
Recursive Languages

i) Union languages

Union of 2 rec. langs. is always recursive. bcz we can construct TM for $L_1 \cup L_2$.

$L_1 \Rightarrow Rec = $ 

$L_2 = Rec = $ 

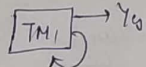
$L_1 \cup L_2$ 

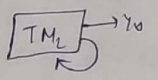
Hence, Rec. languages are closed under Union operation

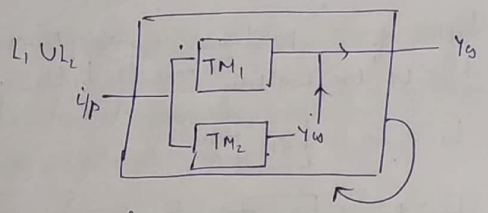
Scanned by CamScanner

2) Union for R.E.L

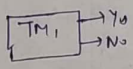
Union of 2 REL's is always R.E.L bcz we can construct TM for $L_1 \cup L_2$

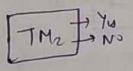
L_1 = R.E.L = 

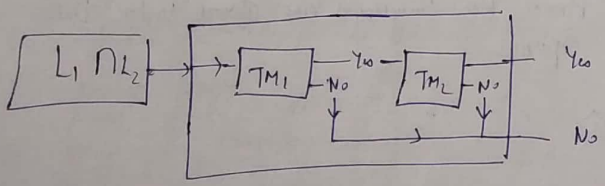
L_2 = R.E.L = 



3) Intersection for Recursive Lang

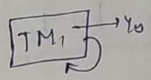
L_1 = 

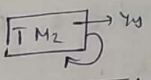
L_2 = 

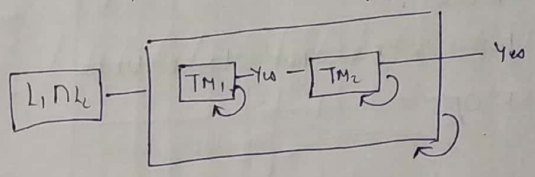


Intersection of 2 Rec. lang's is always Recursive Hence Rec. langs are closed under intersection opn.

4) Intersection for R.E.L

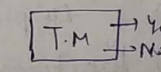
L_1 = R.E.L = 

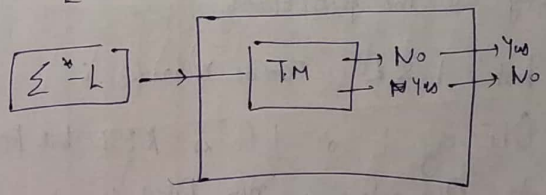
L_1 = R.E.L = 



Intersection of 2 R.E.L's is always R.E.L bcz we can construct T.M for $L_1 \cap L_2$.
Hence R.E.L's are always closed under intersection opn.

5) Complement for Rec. Lang,

L = Rec. Lang = 



T.M

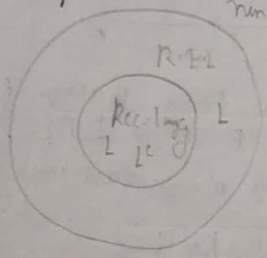
Scanned by CamScanner

Complement of Rec. language is always recursive bcz we can construct TM for $\Sigma^* - L$.

Complement for R.E.L

Complement of R.E.L is R but not Recursive is non-R.E.L.

Hence R.E.L's are not closed under complement opⁿ



(Imp) (Gate)

For a ~~pair~~ pair of complementary languages (L, L^c) following are the possibilities.

i) both L & L^c can be recursive.

ii) One of L or L^c is R.E.L but Recursive then the other is non R.E.L.

iii) both L & L^c can be non-R.E.L

Q) A language and its complement both are R.E.L Then that language is Recursive

Concept - convert problem into language. $D_1 D_2 D_3 \ldots$

$L_1 = \{ \langle D \rangle \mid D$ is a DFA and $L(D) = \phi \}$ Rec lang.

$L_2 = \{ \langle D \rangle \mid D$ is a DFA and $L(D) \neq \phi \}$ → $\Sigma^* - L_1$ = Rec. lang.

$L_3 = \{ \langle G \rangle \mid G$ is a CFG and G is ambiguous $\} \Rightarrow$ R.E.L

$L_4 = \{ \langle G \rangle \mid G$ is a CFG and G is not ambiguous $\} \Rightarrow \Sigma^* - L_3 \Rightarrow$ non R.E.L

$L_5 = \{ \langle D_1, D_2 \rangle \mid D_1, D_2$ are DFA and $L(D_1) = L(D_2) \}$ → Recursive language

$L_6 = \{ \langle D_1, D_2 \rangle \mid D_1, D_2$ are DFA and $L(D_1) \neq L(D_2) \} \Rightarrow$ Rec lang

$L_6 = \Sigma^* - L_5$

Scanned by CamScanner

Operation	Reg. Lang	DCFL	CFL	C.S.L	Rec. lang	REL
1) Union	✓	×	✓	✓	✓	✓
2) Concatenation	✓	✓	✓	✓	✓	✓
3) Complement	✓	×	×	✓	✓	×
4) Intersection	✓	×	×	✓	✓	✓
5) Intersection with R.L	✓	✓	✓	✓	✓	✓
6) Difference ($L_1 - L_2$)	✓	×	×	✓	✓	×
7) Difference with R.L	✓	✓	✓	✓	✓	✓
8) Reversal opn	✓	×	✓	✓	✓	✓
9) Substitution	✓	×	✓	✓	✓	✓
10) Homomorphism	✓	×	✓	✓	✓	✓
11) Inverse Homomorphism	✓	✓	✓	✓	✓	✓
12) Kleen Closure	✓	×	✓	✓	✓	✓
13) Positive closure	✓	×	✓	✓	✓	✓
14) Union with R.L	✓	✓	✓	✓	✓	✓

Q Which of the following is false for under R.E.L

a) Closed under Union
b) Closed under intersection
c) Closed under complement
d) Closed under reverse

Q L_1 is Recursive Language, L_2 & L_3 are R.E.L but not Recursive then which of the following is false

a) $L_2 - L_1 = R.E.L$ REL ∩ Rec ⇒ REL ∩ REL = REL
b) $L_3 \cap L_2 = R.E.L$
c) $L_1 - L_3 = R.E.L$ $L_1 \cap L_3^c$ Rec ∩ non R.E.L = ϕ
d) $L_1 \cup L_2 = R.E.L$ R.EL ∪ REL ⇒ REL

Q L_1 is Regular. L_2 is CFL L_3 is Recursive. L_4 is R.E.L but not Recursive which of the following is false?. REL ∩ REL ∩ REL ∩ REL REL
a) $L_1 \cap L_2 \cap L_3 \cap L_4$ is R.E.L
b) $L_2^c \cup L_3$ is Recursive R ∪ R ∪ Rec = Rec
c) $L_1 \cup L_2^c \cup L_3$ is CFL RLU Rec ∪ REL
⇓
REL ∪ REL = REL
d) None of these

Scanned by CamScanner

Undecidability

A problem is said to be decidable if there exists halting TM or algorithm for that problem.

A problem is said to undecidable if there is no halting TM (or) algorithm for that problem.

Problem Reduction

→ A problem A is reducable to B means we can conclude one problem with the help of other problem.

→ If A reducable to B happens then B is atleast as hard as A.

→ If A reducable to B happens following are the possibilities

a) if A is Undecidable then B is Undecidable.

b) if B is decidable then A is decidable

c) if B is Recursive lang. then A is Recursive lang.

d) if B is R.E.L then A is R.E.L.

e) if A is not rec. then B is not recursive.

f) if A is not R.E.L then B is not R.E.L.

→ Reduction satisfies Transitive property, hence

$A \propto B$

$B \propto C$

---

then $A \propto C$

Gate

Q) X is Rec. lang and Y is R.E.L but not Rec. and $Y^C \propto W$ and $Z \propto X^C$. Which of the following is True ?

a) W and Z are R.E.L

b) W is R.E.L and Z is not Rec.

c) W is not R.E.L and Z is Rec.

d) Both W and Z are R.E.L

Completeness Problem (or) Totality Problem

It means checking whether lang. generated by given grammar is Complete lang. or not.

(or)

lang. accepted by given automata is complete language (or) not.

Scanned by CamScanner

→ This problm is decidable if complemnt opn is closed and emptiness is decidable

Let L_1 be Σ^* ($\Sigma^* - L_1$) must be difficult ie L_1 must be closed under complementation

$\Sigma^* - L_1$ must accept Empty language

Subset Problm

means checking whether language L_1 is subset to language L_2 or not.

Subset problm is decidable if intersection opn is closed and Equivalence problm is decidable

$L_1 \subseteq L_2$

$\equiv$ equivalence

$L_1 \cap L_2$

RL → D
DCFL → UD
CFL → UD
REL → UD

Intersection Empty Problm

$L_1 \cap L_2 = \phi$

means checking whether intersection of given two languages is Empty or not.

This problm is decidable if "$\cap$" opn is closed and Emptiness problm is decidable,

Co-finiteness Problm

means checking whether complemnt of given language is ~~Regular~~ finite or not

$\Sigma^* - L$ = finite ?

RL - D
DCFL - D
CFL - UD
REL - UD

This problm is decidable if complemnt opn is ~~Intersecti~~ closed and finiteness problm is decidable.

Intersection finiteness Problm

$L_1 \cap L_2$ = finite ? → decidable if intersection problem closed & finiteness problem decidable

means checking whether "$\cap$" of given two languages is finite (or) not.

Problms Under Regular language

1) Membership problm : D
2) Emptiness problm : D
3) Finiteness problm : D
4) Equivalence problm : D
5) ~~To~~ Completeness problm : D
6) Subset problm : D
7) Intersection Empty problm : D

Scanned by CamScanner

8) Co-finiteness problem : D

9) Intersection finiteness problem : D

10) Ambiguity problem : D

Problems Under DCFL's

1) Membership problem : D
2) Emptiness problem : D
3) Finiteness problem : D
4) Equivalence problem : D
5) Completeness problem : D
6) Subset problem : UD
7) Intersection Empty : UD
8) Cofiniteness problem : D
9) Intersection finiteness : UD
10) Ambiguity problem : D

Problems Under CFL

1) Membership problem : D
2) Emptiness problem : D
3) Finiteness problem : D
4) Equivalence : UD
5) Completeness : UD
6) Subset : UD
7) Intersection Empty : UD
8) Cofiniteness : UD
9) Intersection finiteness problem : UD
10) Ambiguity problem : UD

Problems Under R.E.L (U) TM

1) Membership UD
2) Emptiness UD
3) Finiteness UD
4) Equivalence UD
5) Completeness UD
6) Subset UD
7) Intersection Empty UD
8) Cofiniteness Problem : UD
9) Intersection finiteness problem : UD
10) Ambiguity problem : UD

Scanned by CamScanner

Halting problem of TM

→ means by reading string x whether the TM halts or not. known as Halting problem

→ Halting problem of TM is Undecidable problem

→ If any Halting problem reducable to any other problem than that problem also Undecidable problem.

Membership Problem

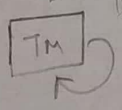
→ means checking whether string x is accepted by given TM or not.

membership problem of TM is Undecidable

Emptiness problem

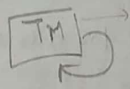
→ It is Undecidable

$L = \{\}$

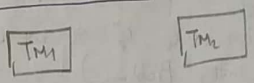


We have designed TM which accepts only Empty lang. (ie it says Yes for ϕ). If any lang other than empty lang is given to this TM it may or may not halt.

Finiteness problem : It is Undecidable



Equivalence problem : It is Undecidable



Whether lang accepted two TM's is same or not

Completeness problem : It is Undecidable

REL not closed under complementation
∴ Emptiness problem is undecidable

Subsets problem : It is Undecidable.

REL is closed under intersection ∴ but equivalence problem is undecidable

Intersection Empty : It is Undecidable

REL closed under intersection ∴ but Emptiness problem is undecidable.

Cofiniteness Problem : It is Undecidable

REL not closed under complementation ∴ finiteness problem is undecidable

Intersection finiteness : It is Undecidable

REL is closed under intersection ∴ but finiteness problem is undefined.

Ambiguity problem : It is Undecidable

Scanned by CamScanner

following problems of TM are decidable (Imp)

i) Checking whether TM having 100 states or not.
We only have to do counting (finite counting)

ii) By reading a string X, whether the TM enters into particular state "q" or not within 100 steps (State Entry Problem)
Here we just have to wait for 100 steps only.

iii) String X is member of given TM or not in within 100 steps.
Decidable bcz we have to wait only for 100 steps.

iv) Whether the TM takes more than 2019 steps on some i/p X or not.
We have to wait for final o/p only for 2019 steps.

Problem	R.L	DCFL	CFL	CSL	Recursive	R.E.L
is $w \in L$	D	D	D	D	D	UD
is $L = \phi$?	D	D	D	UD	UD	UD
is L = finite?	D	D	D	UD	UD	UD
is $L_1 = L_2$?	D	D	UD	UD	UD	UD
is $L = \Sigma^*$	D	D	UD	UD	UD	UD
is $L_1 \subseteq L_2$	D	UD	UD	UD	UD	UD
is $L_1 \cap L_2 = \phi$	D	UD	UD	UD	UD	UD
is $\Sigma^* - L$ = finite?	D	D	UD	UD	UD	UD
is $L_1 \cap L_2$ = finite?	D	UD	UD	UD	UD	UD
is L = regular?	D	D	UD	UD	UD	UD
Ambiguity	D	D	UD	D	D	UD
is $\Sigma^* - L$ same type or not.	D	D	UD	D	D	D
is $L_1 \cap L_2$ same type or not	D	UD	UD	D	D	D

~~The following~~

Scanned by CamScanner

The following problems in programming languages are Undecidable. (Imp)

1) Whether a given program ever produces an o/p ?

2) Whether a given program can loop forever on some i/p or not ?

Gate Q) Which of the following problem is decidable

a) For given Unrestricted grammar G checking whether string belongs to given grammar or not.

b) M is a TM, is $L(M)$ = regular ? ?

c) G_1 is CFG, is $L(G_1) = \Sigma^*$?

d) Is $L(DFA) = L(NFA)$?

Gate Q) Which of the following problem is decidable

a) Whether a given prg. ever produces an o/p

b) M is a TM, is $w \in L(M)$?

c) R is a Regular expression, is $w \in L(R)$?

d) Checking whether given CFG ambiguous or not ?

Post's Correspondence Problem (PCP)

$A = \{ \overset{1}{1}, \overset{2}{10}, \overset{3}{11}, \overset{3}{10} \}$

$B = \{ 111, 10, 0 \}$

We have to find a permutation of sequences A & B which give same string

2113 → this permutation give same string for both the sequence

Q) identify soln for following PCP

① $A = \{ \overset{1}{1}, \overset{2}{1011}, \overset{3}{b} \}$

$B = \{ 111, 10, 0 \}$

2113

$A = 10111110$

$B = 10111110$

② $A = \{ \overset{1}{bbab}, \overset{2}{ab}, \overset{3}{baa}, \overset{4}{b} \}$ 2143 ab bbab b baa

$B = \{ a, abbb, aa, bbb \}$ abbbabbbaa

③ $A = \{ \overset{1}{aaa}, \overset{2}{bb}, \overset{3}{aa} \}$ no solution

$B = \{ aba, bba, b \}$

Scanned by CamScanner

4) A = { ba, ab, a, baa, b }
{ bab, baa, ba, a, aba }

PCP is Undecidable problem, because no algorithm exists to find solution for PCP problem.

Modified PCP (MPCP)

→ In modified post correspondence problem solution is required to start with 1st string of each list.

→ MPCP is reducable to PCP.

→ Both PCP & MPCP are undecidable problems

→ Both PCP & MPCP are decidable problems if list contain one element.

Decidability

Decidability

Modifications of TM

If we modify basic version of TM, expressive power of the TM remains same.

The following are some of the modified versions of TM.→

① Two-way infinite Tape T.M

→ It is a TM in which i/p tape is infinite in both directions

② Multi tape TM

→ TM having one or more i/p tapes is known as multi tape TM

③ → For every Multitape TM we can construct an equivalent single tape TM.

}

Scanned by CamScanner

③ Non-Deterministic Deterministic TM

→ NDTM is a TM in which from the given state, from given tape symbol finite no. of transitions exists.

→ For every NDTM we can construct an equivalent DTM.

④ Universal T.M

It is a TM which takes F.A, PDA, and TM as i/p and represents their behaviour.

(Imp) ⑤ Multistack TM

F.A having 2 or more stacks known as multistack TM.

Q Which of the following is false

a) F.A with one stack is ~~power~~ powerful than F.A

b) F.A with two stack is powerful than F.A with one stack.

c) FA with 3 stacks is powerful than F.A with two stack

d) None of these

Decidability

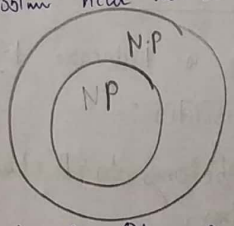
P Class Problem

A problem is said to be in P class if there exists DTM for that problem. (or) deterministic algorithm exists for that problem.

NP - Class Problem

A problem is said to be in NP - class if there is exists NDTM for that problem (or) non-deterministic algorithm exists for that problem

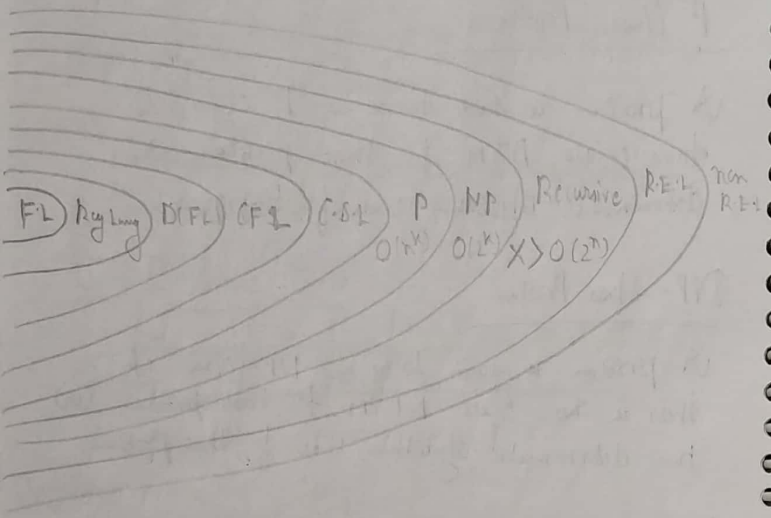
→ Every P-class problem is NP-class problem, but every NP-class problem need not be P-class problem



→ P = NP ? is an Open question ie for every NP class problem if deterministic algorithm (or) DTM is constructed then P = NP happens.

Scanned by CamScanner

→ But still today no one has constructed deterministic algorithm for any NP class problem



NP-Hard Problem

If problem A is reducible to B happens, following are the possibilities :-

i) If B belongs to P class, then A belongs to P class.

ii) If B belongs to NP-class, then A also belongs to NP-Class

iii) If all NP-class problems are reducible to X, then X is known as NP-hard problem

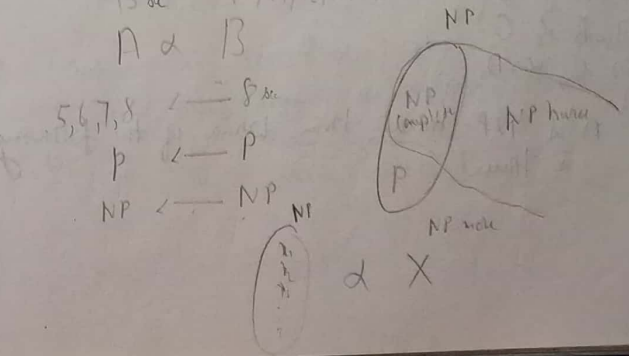
iv) NP hard problem may belong to NP-Class or may not belong to NP class.

v) If any NP hard problem belongs to NP class then it is NP complete problem

vi) All NP-complete problems are NP-hard problems, but all NP-hard problems, need not be NP-complete problems.

vii) NP-complete problem means, problem completely belongs to NP-Class but not belongs to P-Class.

Hence $P \cap NP = \phi$.



Scanned by CamScanner

→ To prove P = NP happens or not ; for at least one NP problem complete problem, DTM has to be constructed.

→ But still today no one has constructed DTM

Q Consider the following reduction 3-SAT α A (3-SAT is NP complete problem) Which of the following is true? 3-SAT α A

a) A belongs to P class
b) A belongs to NP class
c) 'A' is NP complete problem
d) None of these

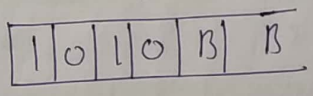
Q Consider the following 3 reductions.
i) A α B
ii) B α C
iii) C α D

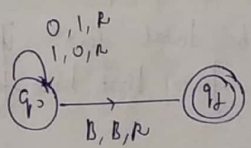
A is NP-hard then which of the following is true?

a) 3-SAT α D ⇒ D is NP-complete
b) A α 3-SAT ⇒ A is P class
c) D α 3-SAT ⇒ D is NP complete
d) None

T.M as O/p Generator

Q Construct the TM that produces 1's complement of given binary number as o/p ?

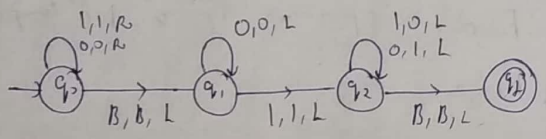




Q Construct the TM that produces 2's complement of given binary number as o/p. (Assume we are using 2-way infinite tape TM)

Scanned by CamScanner

B	1	0	1	0	1	0	0	B	B



→ functions performed by TM are 2 types

i) Partial function

ii) Total function

Total function

A fun^n is said to be total fun^n if there exists a TM for that fun^n that halts for all function values.

→ Total functions are similar to Recursive languages.

Partial functions

A fun^n is said to be partial fun^n if there exists a TM for that fun that halts for some fun^n values, may not halts for some fun^n values

→ Partial fun^n's are similar to R.E.L.

TM as language Generator (or) Enumerator

→ Enumerator produces o/p without taking any i/p.

→ Enumerator is a predefined m/c or program.

→ Whenever we turn on Enumerator, it produces languages as o/p

→ The languages generated by enumerator are exactly same language accepted by TM. Hence Enumerator generated languages are Recursive language, and R.E.L.

Scanned by CamScanner

→ If the strings produced by enumerator are in proper order, then that language is known as recursive language.

→ If the strings produced by enumerator are not in proper order, that language is known as R.E.L.

Q If the strings produced by enumerator are arranged in dictionary order that language is known as Recursive

(2005)

Q Let $L = \begin{cases} \phi & \text{if } P = NP \\ (a+b)^* & \text{if } P \neq NP \end{cases}$

then L is ?

a) Regular

b) CFL but not Regular

c) Recursive but not CFL

d) We can't say until P = NP problem is ~~still~~ decided.

14/10/18

Compiler Design (3-5)

Course

10% → Lexical Analysis

60% → Parsing

30% → Syntax Directed Translation
Intermediate Code Generation
Runtime Environments

Definition of Compiler

Compiler is a program that translates a program written in one language into an equivalent program of another language. And also detects errors present in a program.

Compiler detects 3 type of errors known as

i) Lexical Errors
ii) Syntax Errors
iii) Semantics Errors

Scanned by CamScanner

Lexical Analysis Phase

→ This a program that takes high level language as i/p and produces tokens as o/p. Also detects Lexical errors.

Syntax Analysis Phase

→ This a program ~~that~~ that takes tokens as i/p and produces parse tree as o/p. Also detects Syntax errors. present in the program

Semantic Analysis Phase

→ Takes parse tree as i/p and produces annotated parse tree as o/p. And also detects Semantic errors present in the program

→ First 3 phases of the compiler are known as analysis part of compiler because they are used for error detections only

Intermediate Code gneration Phase

→ It is a prog. that translates high level language into Three address code.

→ Advantage of gnerating intermediate code is to perform optimizations

Code Optimization Phase

→ It is a program that eliminates unnecessary information present in the high level language. so that time and space of target m/c is reduced

Code Generation Phase

→ It is a program that translates optimized intermediate code into assembly language.

→ Last 3 phases of Compiler are known as translation part of compiler (or) Synthesis phase

Scanned by CamScanner

Symbol Table

→ It is a data structure that contains information about identifiers and constants present in the program.

Error Handler

→ It is a program that contains information about errors present in the the high level language.

→ If any phase of the compiler detects error that information is send to error handler

Front End of Compiler

→ Front End of compiler means phases of compiler which completely depends on source language and independent of target m/c.

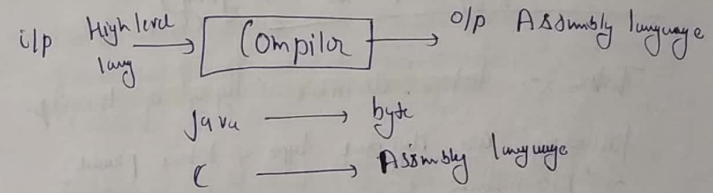
Front End phase includes :-

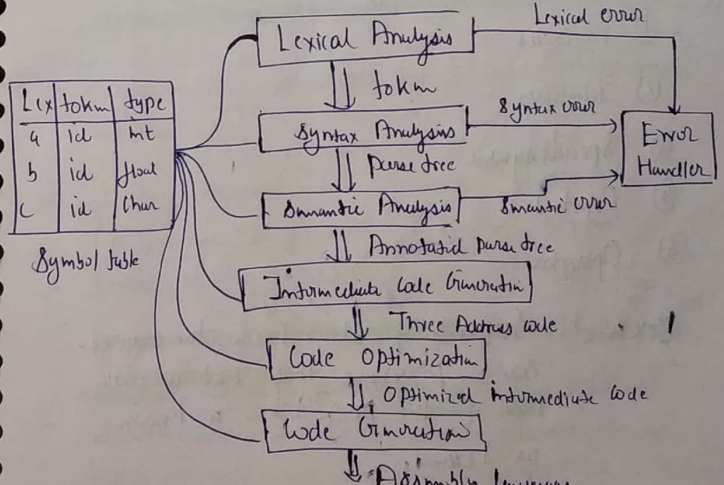
1) Lexical Analysis
2) Syntax Analysis
3) Semantic Analysis
4) Intermediate Code Generation

Back end of Compiler

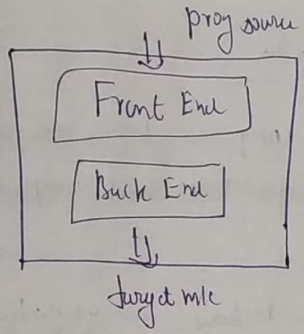
→ The phases of compiler which completely depends on target m/c and independent of source language :

→ Back end phases includes code generation and m/c dependent optimization





Scanned by CamScanner



Lexical Analysis Phase (Designing)

Token :- token describes category of the i/p.

following are different type of tokens present in programming language :-

1. Keywords
2. Identifiers
3. Special symbol
4. Constant
5. Operator.

Lexeme :- Sequence of characters in the ~~source~~ source program that matches with rules of the ~~prog~~ token is known as lexeme.

→ Any sequence of characters is not a lexeme then Lexical error will be generated.

→ Lexical Analyzer reads sequence of characters as i/p and groups characters into meaningful Lexemes known as tokens.

→ Lexical Analyzer eliminates all comment lines in the program.

→ Lexical Analyzer eliminates all white space characters present in the program which include

1) blank 2) tab 3) new line

→ To design Lexical analysis phase regular expression and finite automata is used.

→ Corresponding to every token finite automata is constructed.

→ If any sequence of characters matching with FA, token is produced. otherwise ~~Syntax~~ Lexical error is generated

→ If any sequence of characters matches with identifier automata, then search op^n is performed with keyword table. If search is success keyword token is returned otherwise identifier token is returned

Scanned by CamScanner

→ If any sequence of characters matches with whitespace automata or comment automata then no token is produced so that Lexical analyzer eliminates all whitespace characters and comment lines in present in the token

Drawback of Lexical Analysis

→ Lexical analyzer can not detect Syntax and Symantic errors present in the program

Q Identify no. of tokens produced by "C" compiler for the following program statements.

① for (i=1; i<100 ; i++) → 13

② a = b + c * d ; — 8

③ printf ("Compiler"); — 5

④ int a, b; /* GATE */ — 5

⑤ a = b + c ; /* Examination */ — 6

⑥ main()
{
int x, y;
~~// /* gate */ out z;~~
// /* gate */ out z;
x = /* exam */ ;
y = 20;
}

21

Q Which of the following C program statements having lexical error

① int a, b ; — 5 — no Error

② if (a > b) — 6 — no Error

③ for (i=1; i<100 ; i++) — 13 — no Error

④ a = b + c ; * d ; — 9 — no error

⑤ ~~int /* compiler */ t~~ ~~int~~

⑥ in /* compiler */ t x; — 4 — no error

⑦ a = /* Design */ b + c; — 6 — no error

⑧ int a = "GATE"; — 5 — no error
constant

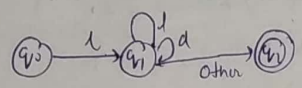
Scanned by CamScanner

g) int a = "Exam"; missing " Error

h) char b = 'a'; missing ' Error

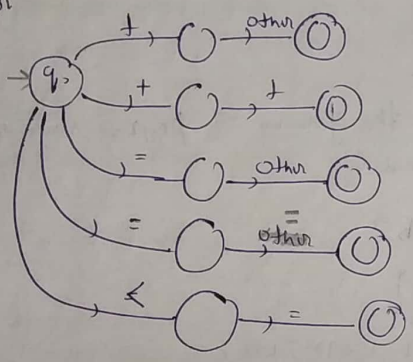
Misc
if
for
int

ID FA

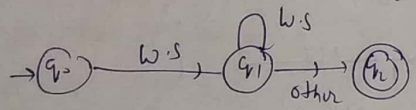


l = a b c ... z l = letters
d = 0, 1, ... 9 d = digits

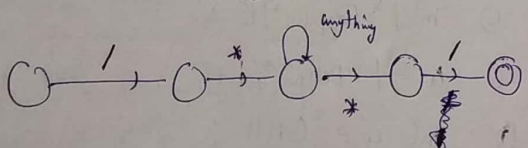
Operators



WS = blank / tab / new line



Comment



Q) What is the type of error present in the following C program?

```
main()
{
    int i;
    for (i=1, i<10, i++)
}
```

→ compile time error

a) Lexical Error

b) Syntax Error

c) Lexical and Syntax error.

d) No compile time error

Q) Identify no. of tokens produced by lexical analyzer for the following C programs.

① a = b; 4

② a = = b; 4

③ a = b + c; 6

④ i++; 3

⑤ a <= b; 4

Scanned by CamScanner

Syntax Analysis Phase

Syntax :- Order of arranging tokens is known as syntax

Parsing :- Checking whether string is is member of given grammar (or) not. known as parsing.

Syntax Analyzer (or) Parser :- It is a program that takes tokens and Context Free Grammar as i/p and verifies tokens are member of CFG or not.

-> If tokens are members, parse tree is the o/p. Otherwise Syntax error is generated

→ To design parser 2 types of algorithm exists

i) Top down parsing algorithm

ii) Bottom Up parsing algorithm

→ Parser is a Deterministic PDA

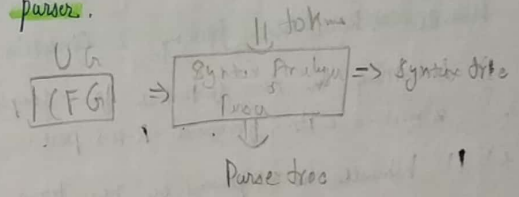
→ Syntax of programming languages are represented by using CFG's.

→ There are 2 types of CFG

i) Ambiguous CFG

ii) Unambiguous CFG

→ Unambiguous CFG's are used to design parser.



Bottom Up Parsing

→ Bottom Up parser construct parse tree starting from the given string proceeds towards starting symbol of the grammar.

→ 2 type of algorithms exists in bottom up parsing

1) LR parsing algorithm

2) Operator precedence parsing algorithm

Scanned by CamScanner

Handle :- Handle is a substring in the sentential form, that completely matches with R.H.S part of grammar production and it is replaced by corresponding L.H.S non-terminal.

→ Every matching thing is not a handle

→ Handle should have following 2 conditions

1) Substring in the sentential form completely is matches with grammar R.H.S part.

2) If handle is replaced by non-terminal it should leads to starting symbol of grammar

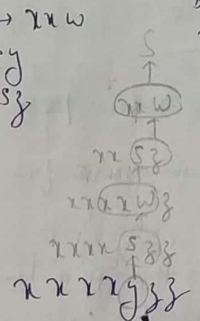
→ Detection of proper no of the handles is difficult through bottom up parsing.

→ Bottom up parsers uses Right Most Derivation in Reverse Order while constructing parse tree.

Q Identify total no. of handles detected by Bottom up parsers for following i/p strings by using given grammar?

1 $S \to xxW$
$S \to y$
$W \to Sz$

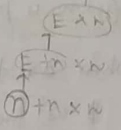
String
$xxxxyzz$



handles = 5

Sol:
$xxxxyzz$

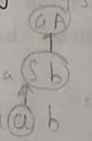
2) $E \to E+n \mid E*n \mid n$ String : $n+n*n$



handle = 3

③ $S \to aA$
$S \to a$
$A \to Sb$

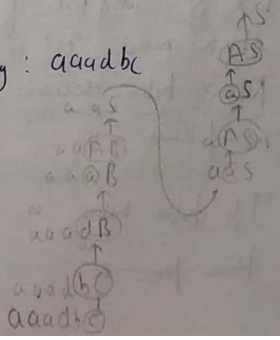
String : aab



handles = 3

④ $S \to AS \mid AB$
$A \to a$
$B \to bC \mid dB$
$C \to c$

String : $aaadbc$



handles = 9

Scanned by CamScanner

LR parsing Algorithm

→ LR parser is a shift reduce parser.

→ Shift action means push symbols from i/p buffer to stack

→ Reduce action means if there is a handle in the stack it is replaced by corresponding non-terminal.

→ for valid syntax, accept action is produced.

→ for invalid syntax, error action is produced.

→ To perform LR parsing tokens are taken into i/p buffer. along with $ as end marker

→ LR parser takes decision regarding shift and reduce actions by visiting the parsing table

→ LR parsing table is a deterministic FA

→ Let S is state of DFA on the top of the stack and "a" is lookahead symbol. Then parser takes decision from the parsing table as follows:

① If table entry is A[S,a] = shift j

~~then push a~~

① A[S,a] = shift j

a) push "a" into stack

b) push "j" into top of stack

c) Increment look ahead symbol

② If table entry is A[S,a] = "S on 'a'" is reduce A→B

= Reduce A→B

A[S,a] = Reduce A→B

(1) pop 2*|B| (ie length of B symbol from stack)

(2) push A into stack

(3) Then push GOTO(i, A) into Top of stack

where i = previous stack on the stack

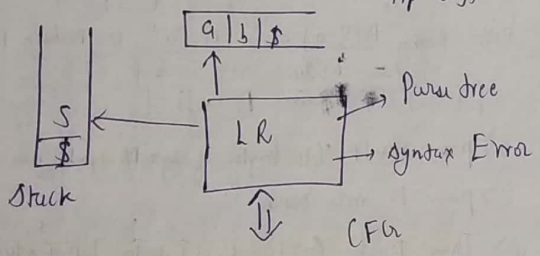
③ Table Entry A[S,a] = Accept ; i/p string is valid (ie valid syntax)

④ Table Entry A[S,a] = blank ; then there is syntax error and it to error handler.

→ Time complexity of LR parsing algorithm is O(n) where n = "length of string"

Scanned by CamScanner

Q how many shift action and Reduce actions are taken by the parser to parse the i/p ~~AB~~ "ab" by using following grammar and parsing table.



LA

	a	b	$	Action
I_0	S_3			
I_1			accept	
I_2		S_5		
I_3		r_2		
I_4			r_1	
I_5			r_3	

Goto (NT)

S	A	B
1	2	
		4

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b$

String : "ab"

Stack	input	Action
	ab $	$[0,a] = S_3$
$ 0 a 3	b $	$[3,b] = R$ $A \rightarrow a$
$ 0 A 2	b $	$[2,b] = S_5$
$ 0 A 2 b 5	$	$[5,\$] = R$ $B \rightarrow b$
$ 0 A 2 B 4	$	$[4,\$] = R$ $S \rightarrow AB$
$ 0 S 1	$	$[1,\$]$ = Accept

# of shifts = 2 # of reduce = 3

Q How many shift actions and Reduce actions are taken by LR parser to parse the i/p id + id x id by using following grammar and parsing table,

Scanned by CamScanner

Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow id$

Parsing table

	Action				GOTO
	id	+	*	$	E
I_0	S_2				1
I_1		S_3	S_4	accept	
I_2		r_3	r_3	r_3	
I_3	S_2				5
I_4	S_2				6
I_5		r_1	S_4	r_1	
I_6		r_2	r_2	r_2	

id	+	id	*	id	$

Stack

$	0	id	2	+	3	id	2	*	4	id	2
			E	1			E	8			E
			E	1			E	8			

↑ accept

# of r = count all non-terminals

# of s = count all terminals

→ LR parsing table is deterministic F.A

→ To construct LR parsing table 4 methods exists known as :-

i) LR(0) ii) SLR(1) iii) CLR(1)

iv) LALR(1)

LR(0) parsing table Construction

→ Construct Augmented grammar by adding a production "S' → S"

→ Advantage of adding new production is to get accept action in the parsing table

→ Construct LR(0) items DFA by using closure() function and goto() function. LR(0) item is nothing but any CFG production having "." on RHS part of grammar.

→ Even though we can not construct DFA for CFG we can construct DFA for LR(0) items.

→ The LR(0) items $A \rightarrow X.YZ$ gives the meaning X present in the stack and YZ present in the i/p. Hence parser has to perform shift action.

Scanned by CamScanner

→ The RR LR(0) item $A \to XYZ.$ gives the meaning XYZ present in the stack. Hence parser can perform reduce action.

→ Total no of LR(0) items corresponding to given grammar are calculated by using closure() fun^n and goto() fun^n

→ Task of closure() fun^n is if in any LR(0) item "." followed by non-terminal exists (ie $A \to \beta.C$) then that non-terminal production is added and places "." in the first position.

→ ~~Task of~~ Ex:- $S \to A.B$
$B \to .b$

→ Task of goto() fun^n is It moves "." one position towards right side.

→ LR(0) items DFA is converted into transition table notation ie LR(0) parsing table.

Q1 Construct LR(0) parsing table for the following grammar

Q2 Verify given grammar is LR(0) grammar or not.

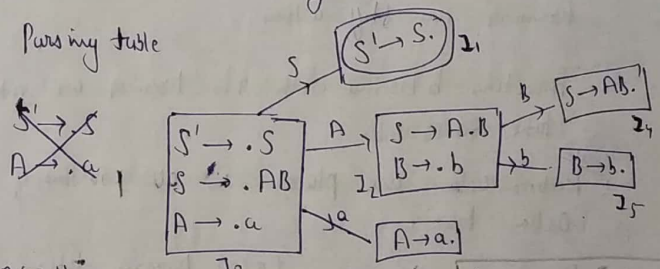
$S \to AB.$
$A \to a$
$B \to b$

given grammar is LR(0) bcz there are no S/R (or) R/R conflict

① Augmented grammar $S' \to S$

② LR(0) items DFA → closure()
→ goto()

③ Parsing table



terminal = shift
Non terminal = goto()

LR(0) items
$S \to .AB$
$S \to A.B$
$S \to AB.$
$A \to .a$
$A \to a.$
$B \to .b$
$B \to b.$

closure()
$S \to A.B$
$B \to .b$

goto()
$A \to .a$ —a→ $A \to a.$

Scanned by CamScanner

State (L.A)	Action			GOTO (NT)		
	a	b	$	S	A	B
I_0	S_3			1	2	
I_1			Accept			
I_2		S_5				4
I_3	r_2	r_2	r_2			
I_4	r_1	r_1	r_1			
I_5	r_3	r_3	r_3			

GOTO means table
goto means look in DFA

→ In any LR(0) items, transitions labelled with terminals are shift actions

→ Transitions labelled with non-terminals are goto GOTO table GOTO table

→ Reduce actions are placed in all positions of action table.

→ In Any given grammar LR(0) parsing table, contains Shift/Reduce conflict or Reduce/Reduce conflict then that grammar is not LR(0) grammar.

→ All LR(0) grammars are unambiguous grammars

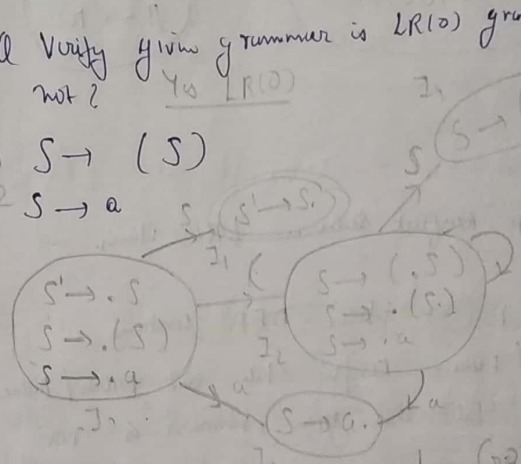
→ Any ambiguous grammar LR(0) parsing table should ~~contains~~ ~~conflicts~~ contain conflict

Q Verify given grammar is LR(0) grammar or not?

Yes LR(0)

1 $S \rightarrow (S)$

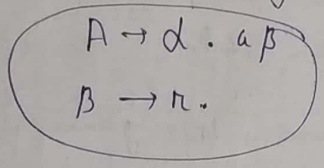
2 $S \rightarrow a$



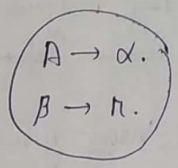
	a	(	)	$	S
I_0	S_3	S_2			1
I_1				accept	
I_2	S_3	S_2			4
I_3	r_2	r_2	r_2	r_2	
I_4			S_5		
I_5	r_1	r_1	r_1	r_1	

Scanned by CamScanner

Shift/reduce conflict



Reduce/Reduce Conflict

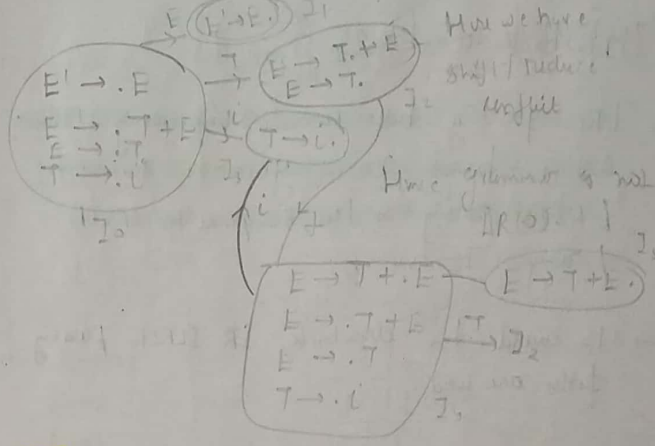


Above are conflict states in LR(0) items.

↳ If LR(0) items DFA contains Shift/reduce conflict state (or) reduce/reduce conflict state then given grammar is not LR(0) grammar.

Q Verify following grammar is LR(0) grammar or not

E → T+E
E → T
T → i



NOTE

The LR(0) item (A → .ϵ) = (A → ϵ.)

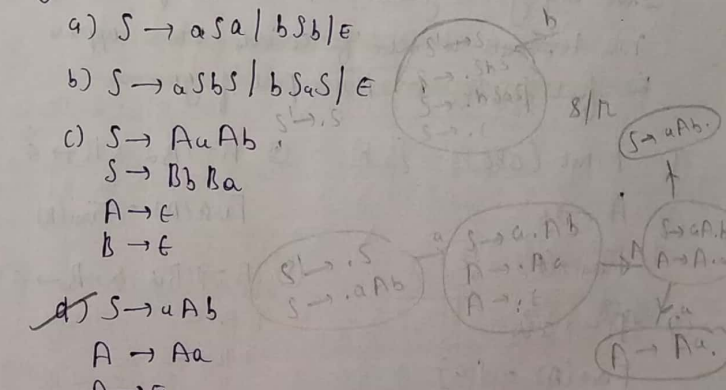
Q Which of the following grammar is LR(0) grammar

a) S → aSa | bSb | ϵ

b) S → aSbS | bSaS | ϵ

c) S → AaAb
S → BbBa
A → ϵ
B → ϵ

d) S → aAb
A → Aa
A → ϵ



Scanned by CamScanner

Drawback of LR(0)

→ LR(0) parsing table contains unnecessary reduce actions, hence no. of grammars suitable for LR(0) parsing is less compared to SLR(1) parsing. ~~To cure~~

→ To avoid this drawback, ~~SR~~ SLR(1) parsing tables are used.

SLR(1) Parsing Tables

First Set Analysis

In any given grammar production first( ) of non-terminal is set of terminals which appears in first position on R.H.S part of grammar.

① First(abc) = {a}.

② A → ∈
First(A) = {∈}

③ A → aα
First(A) = {a}

④ A → Bα & B → ∈
First(A) = First(B)

⑤ A → Bα & B → ∈
First(A) = {First(B) − ∈} ∪ First(α)

Q Calculate First Set of each non-terminal for the following grammars.

① S → AB
A → a/b
B → c/d

S	{a,b}
A	{a,b}
B	{c,d}

② S → AB
A → ∈
B → b

S	{b}
A	{∈}
B	{b}

③ S → AB
A → ∈
B → b/∈

S	{b,∈}
A	{∈}
B	{b,∈}

④ S → ACB/Cb/Ba
A → da/BC
B → g/∈
C → h/∈

S	{d,g,h,∈,b,a}
A	{d,g,c,∈}
B	{g,∈}
C	{h,∈}

⑤ S → aAb/bAaB/∈
A → S
B → S

S	{a,b,∈}
A	{a,b,∈}
B	{a,b,∈}

Scanned by CamScanner

Q Consider the following grammar productions

$A \rightarrow A_1 A_2 A_3$ Where First(A_1) contains 5 elements, First(A_2) contains 3 elements & First(A_3) contains 4 elements. And all First( ) sets contains ϵ and all elements are different. How many elements are there in First(A)?

Sol: $A \rightarrow A_1 A_2 A_3$

$4 + 3 + 3 + \epsilon = 10$

Follow Set Analysis

Follow of a non-terminal is set of terminals which appears immediately R.H.S part of that non-terminal in some sentential form

X does not produce ϵ directly or indirectly

① Follow(S) = { $ }

② $A \rightarrow \alpha B X$ & $X \not\rightarrow \epsilon$

Follow(B) = First(X)

③ $A \rightarrow \alpha B$

Follow(B) = Follow(A)

④ $A \rightarrow \alpha B X$ & $X \rightarrow \epsilon$

Follow(B) = {First(X) − ϵ} ∪ {Follow(A)}

NOTE: Follow set does not include ϵ

Q Calculate follow of each non-terminal for the given grammar

① $S \rightarrow Aa\,A\,b \mid Bb\,Ba$

$A \rightarrow \epsilon$

$B \rightarrow \epsilon$

S	{ $ }
A	{ a, b }
B	{ b, a }

② $E \rightarrow TE'$

$E' \rightarrow +TE' / \epsilon$

$T \rightarrow a$

E	{ $ }
E'	{ $ }
T	{ +, $ }

Follow(T) = First(E')

= { +, $ } → take all the elements except ϵ, substitute "ϵ" in place of E' then calculate again

Scanned by CamScanner

③ $S \to A(B \mid Cb \mid Ba$
$A \to da \mid BC$
$B \to g \mid \epsilon$
$C \to h \mid \epsilon$

S	{$}
A	{h, g, $}
B	{a, $, h, g}
C	{b, g, $, h}

④ $S \to aAbB \mid bAaB \mid \epsilon$
$A \to S$
$B \to S$

S	{$, a, b}
A	{b, a}
B	{$, a, b}

⑤ $S \to iEtSS' \mid a$
$S' \to eS \mid \epsilon$
$E \to b$

S	{$, e}
S'	{$, e}
E	{t}

First

S	{i, a}
S'	{e, ε}
E	{b}

15/10/18

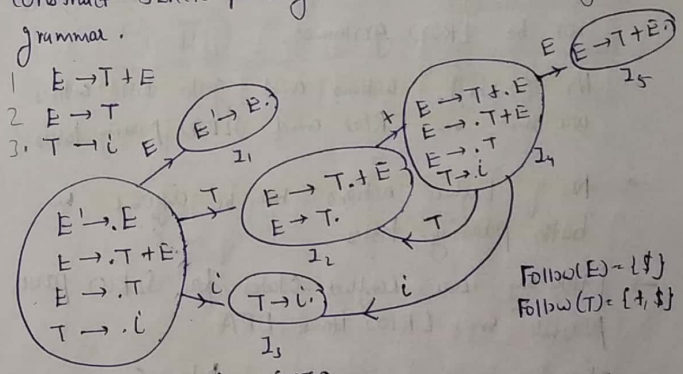
Simplified LR(1) [SR SLR(1)]

Steps to Construct SLR(1)

1) Construct Augmented grammar.
2) Construct LR(0) items DFA.
3) Calculate Follow set of each non-terminal from the given grammar.
4) Reduce actions are placed in the parsing table based on Follow set value of non-terminal.

Q Construct SLR(1) parsing table for the following grammar.

1 $E \to T + E$
2 $E \to T$
3. $T \to i$



Follow(E) = {$}
Follow(T) = {+, $}

	Action			GOTO	
	i	+	$	E	T
I_0	S_3			1	2
I_1			accept		
I_2 {$}		S_4	r_2		
I_3 {+, $}		r_3	r_3		
I_4	S_3			5	2
I_5 {$}			r_1		

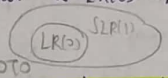
Scanned by CamScanner

NOTE:

→ Any grammar, SR: SLR(1) parsing table contains shift/reduce (or) reduce/reduce conflict that grammar is not SLR(1) grammar.

→ Any given ambiguous grammar ~~is~~ SLR(1) parsing table contains either shift/reduce conflict (or) reduce/reduce conflict.

→ Every LR(0) grammar is SLR(1) grammar, but every SLR(1) grammar ~~is LR(0)~~ need not be LR(0) grammar.

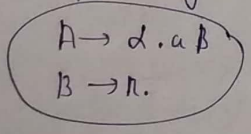


→ No. of shift actions and ~~Goto~~ GOTO table entries are same in LR(0) and SLR(1) parsing tables.

→ No. of reduce actions may be different in both parsing tables.

→ following are conflict states for SLR(1) parser present in LR(0) items DFA.

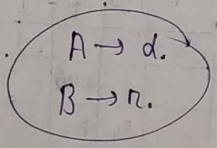
S/R Conflict



if $a \in Follow(B)$

i.e shifting terminal present in the follow of reduce non-terminal

R/R Conflict



$Follow(A) \cap Follow(B) \neq \phi$

Q Which of the following grammar is not SLR(1) grammar.

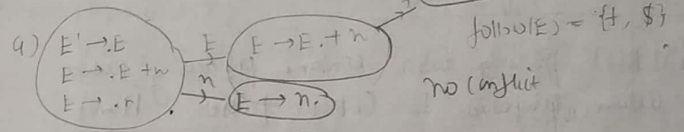
a) $E \to E+n$
$E \to n$

b) $S \to Aa$
$S \to dc$
$A \to d$

c) $S \to Aa$
$S \to Bb$
$A \to d$
$B \to d$

d) $S \to AaAb$
$S \to BbBa$
$A \to E$
$B \to E$

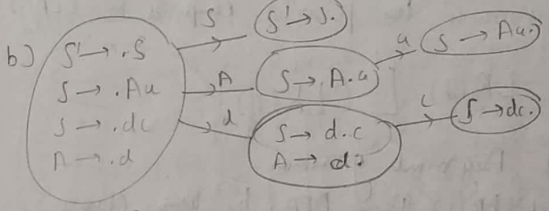
a)



follow(E) = {+, $}

no conflict

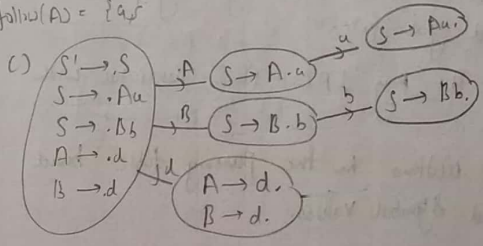
b)



no conflict

follow(A) = {a}

c)

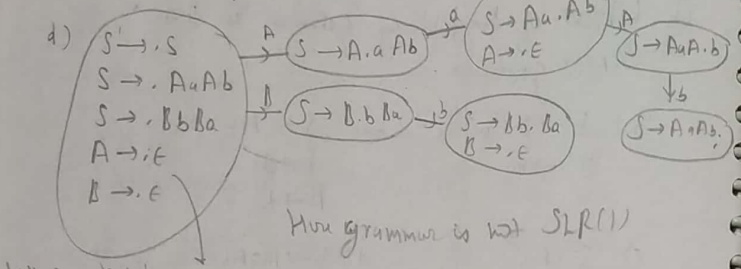


no conflict

Follow(A) = {a}

Follow(B) = {b}

Scanned by CamScanner



Here grammar is not SLR(1)

follow (A) = {a, b}

follow (B) = {a, b}

R/R conflict ∴ follow(A) ∩ follow(B) ≠ ϕ.

Drawback of SLR(1)

→ SLR(1) parsing table contains unnecessary reduce actions compared to CLR(1) parser. Hence SLR(1) parser is less powerful compared to CLR(1)

Canonical LR(1) [CLR(1)]

→ Construct Augment grammar

→ Construct LR(1) Items DFA. by using closure function and goto fun.

→ LR(1) item is any LR(0) item associated with lookahead.

→ Place reduce actions in the parsing table based on lookahead Symbol values.

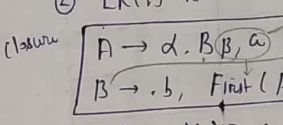
Q1 Construct CLR(1) parsing table for the following grammar

Q2 Verify given grammar is CLR(1) grammar & not.

S → CC
C → aC
C → d

Sol: ① AG

② LR(1) Item DFA



We got this from A → α.Bβ, a so to get its lookahead go to production A and find First (βa)

A → X.YZ, a

LR(1) items DFA

no of shift actions = 8

no of goto = 5

Scanned by CamScanner

	Action			GOTO	
	a	d	$	S	C
I_0	S_3	S_4		1	2
I_1					
I_2	S_6	S_7			5
I_3	S_3				8
I_4	π_3	π_3			
I_5			π_1		
I_6	S_6				9
I_7			π_3		
I_8	π_2	π_2			
I_9			π_2		

Conflict State for CLR(1) parser

S/R Conflict

$A \to \alpha . a\beta, \{t_1\}$

$B \to \pi . \quad , \{t_2\}$

if $a \in t_2$

R/R Conflict

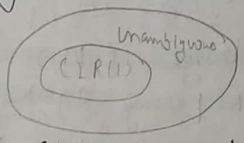
$A \to \alpha ., \{t_1\}$

$B \to \pi ., \{t_2\}$

if $\{t_1\} \cap \{t_2\} \neq \phi$

→ LR(1) items DFA contains S/R conflict state (or) R/R conflict state then the given grammar is not CLR(1) grammar

→ All CLR(1) grammars are unambiguous grammars but all unambiguous grammars need not be CLR(1) grammar

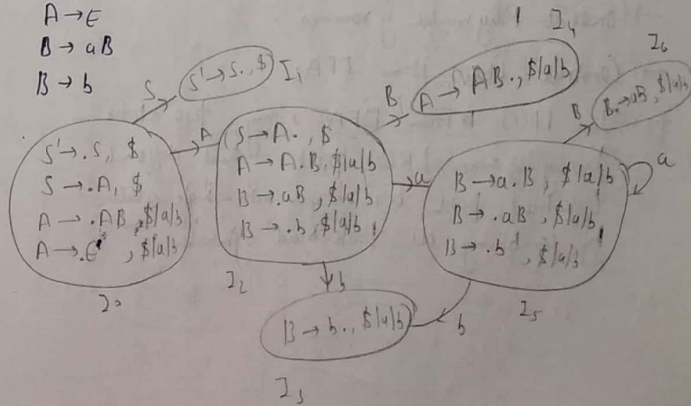


→ Any ambiguous grammar, CLR(1) parsing table contains S/R conflict (or) R/R conflict

Q Verify the following grammar is CLR(1) grammar or not

Here given G is CLR(1) grammar

$S \to A$
$A \to AB$
$A \to \epsilon$
$B \to aB$
$B \to b$



Scanned by CamScanner

Drawback of CLR(1) Parser

→ CLR(1) parser table size is large than LALR(1).
& Hence LALR(1) parsing tables are used in LR parsing algorithms.

→ Even though CLR(1) is powerful than LALR, but LR parsing algorithm using LALR tables.

→ parser generator tool YACC also produces LALR(1) parsing tables

YACC → Yet Another Compiler Compiler

Lookahead LR(1) [LALR(1)]

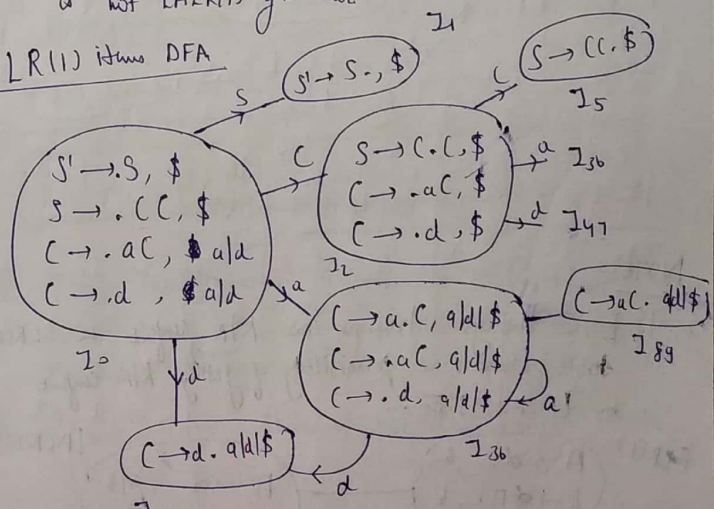
→ Construct Augmented grammar.
→ Construct LR(1) items DFA.
→ In LR(1) Item DFA, any two states having same LR(0) part and different Lookahead part are merged into single state by combining all lookahead symbols.

Q Verify the following grammar is LALR(1) grammar or not?

S → CC
C → aC
C → d

NOTE:
→ The conflict states for LALR(1) parser is same as CLR(1) parser conflict state
→ If the modified DFA contains either S/R conflict state (or) R/R conflict state then that grammar is not LALR(1) grammar

LR(1) items DFA

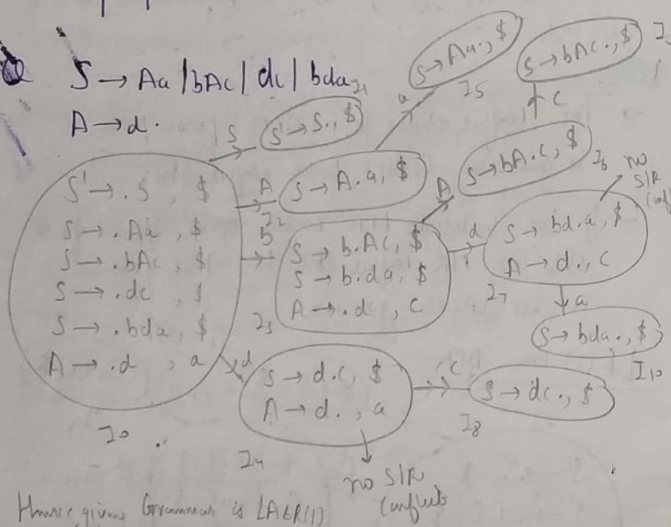


Scanned by CamScanner

	a	d	$
I_{89}	r_2	r_2	r_2
I_5			r_1
I_{47}	r_3	r_3	r_3

Q. $S \to Aa \mid bAc \mid dc \mid bda$

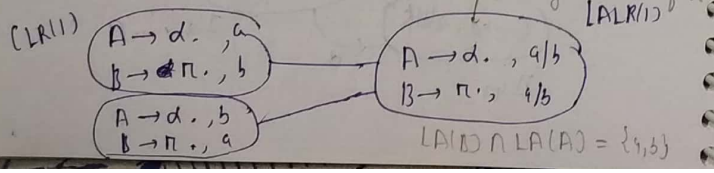
$A \to d$.



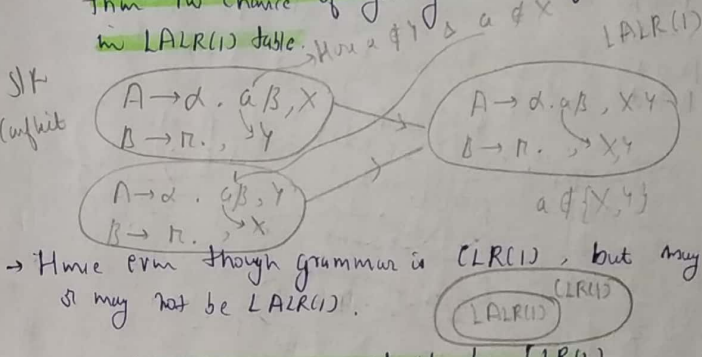
Hence given Grammar is LALR(1)

NOTE:

1) Even though there is no R/R conflict in CLR(1) table there is a possibility of getting R/R conflict in LALR(1) table.



2) If there is no S/R conflict in CLR(1) table then no chance of getting S/R conflict in LALR(1) table.



→ Hence even though grammar is CLR(1), but may or may not be LALR(1).

→ Every LALR(1) grammar should be CLR(1) grammar but every CLR(1) grammar need not be LALR(1) grammar

# of states in LR parsing methods

$$LR(0) = SLR(1) = LALR(1) < \doteq CLR(1)$$

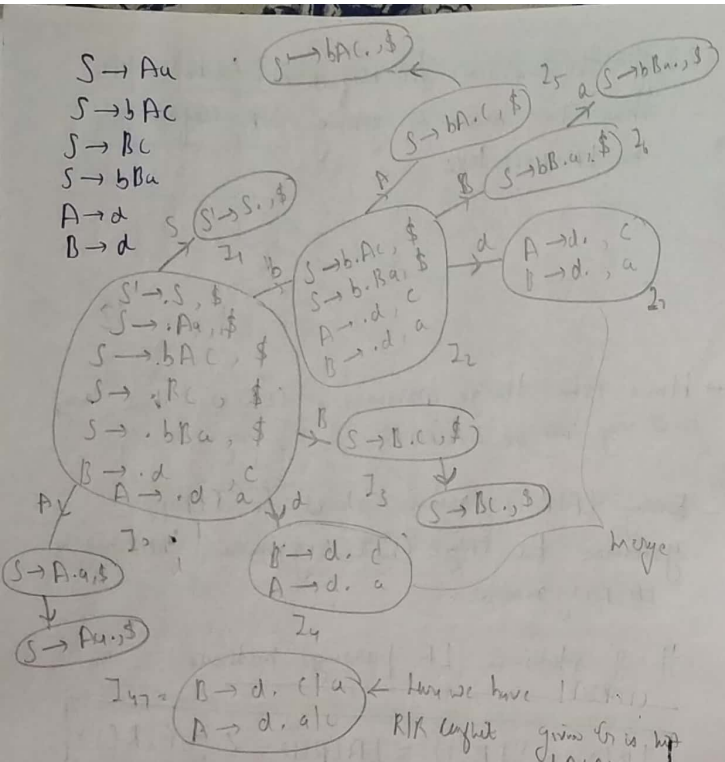
Q. Verify the following grammar is CLR(1) grammar or not. and LALR(1) grammar or not

$S \to Aa$

Q. Find no of states of SLR(1), CLR(1) and LALR(1) parsing tables

Scanned by CamScanner

$S \to Aa$
$S \to bAc$
$S \to Bc$
$S \to bBa$
$A \to d$
$B \to d$



Here given grammar is CLR(1) but not LALR(1)

2) # of states in SLR(1) = LALR(1) <= CLR(1)

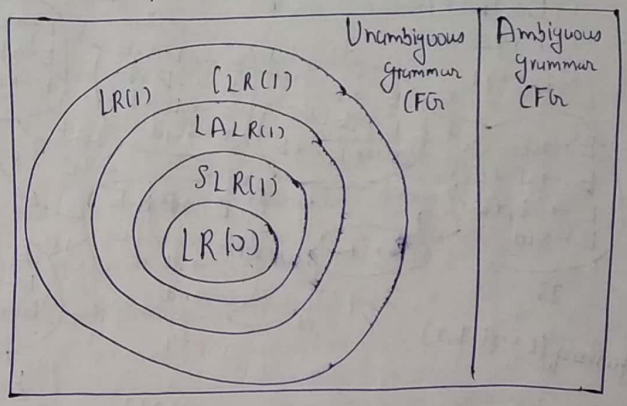
CLR(1) = 13

SLR(1) = LALR(1) = 12

Power of Parser

→ # of grammars suitable for particular parsing method known as power of that parser.

→ By Default LR parser is CLR(1) parser



CLR(1) ⊃ LALR(1) ⊃ SLR(1) ⊃ LR(0)

Scanned by CamScanner

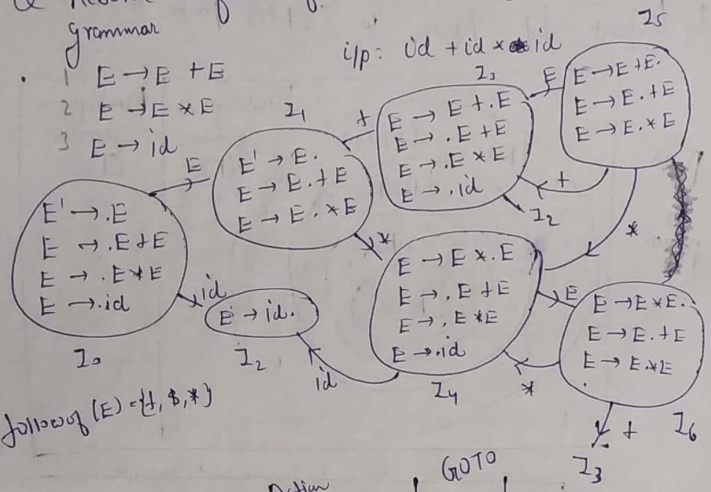
16/10/18

Ambiguous Grammar in LR parsing

Q Resolve conflicts from the following ambiguous grammar

1. $E \to E + E$
2. $E \to E * E$
3. $E \to id$

i/p: id + id * id



Follow(E) = {+, $, *}

	Action				GOTO
	id	+	*	$	E
I_0	S_2				1
I_1		S_3	S_4	accept	
I_2		π_3	π_3	π_3	
I_3	S_2				5
I_4	S_2				6
I_5		S/π_1	S_4/π_1	π_1	
I_6		S/π_2	S_4/π_2	π_2	

NOTE: The shift/reduce conflict at I_5 state on * is resolved by giving priority for shift action in order to make * as higher precedence than +.

→ S/R conflict at I_5 on "+" is resolved by giving priority for reduce action in order to make "+" left-associative

→ S/R conflict at I_6 state on "+" is resolved by giving priority for reduce action in order make "*" as higher precedence than "+".

→ S/R conflict at I_6 state on "*" is resolved by giving priority for reduce action in order to make "*" as left associative.

→ For general ambiguous grammars S/R conflict is resolved by giving priority for shift action over reduce action.

→ For general ambiguous grammars R/R conflict is resolved by giving priority for 1st reduce action over 2nd reduce action.

→ The parser generator tool YACC also resolves conflicts in the same manner.

Scanned by CamScanner

Gate 2010
IITB

following ambiguous grammar is fed to YACC tool. What precedence and associativity rules parser realizes. And what is the value of the expression 3*2+1. (precedence and associativity of operators are not known to YACC)

$E \rightarrow E+E$
$E \rightarrow E*E$
$E \rightarrow id$

ambiguous grammar is fed to the YACC

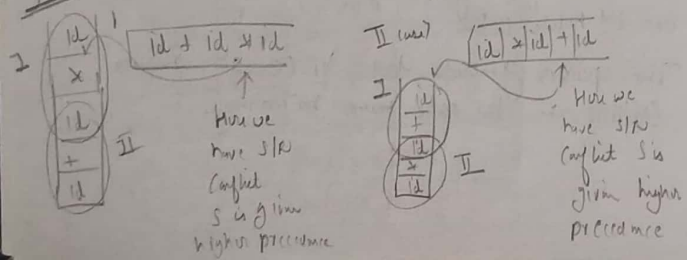
a) * is higher precedence than + & both operators are left associative, Expn. value is 7

b) + is higher precedence than * & both operators are right associative, Expr. value is 9.

c) both the operators are having equal precedence right associative expression value is 9.

d) none of these

Sol:- Parser by default S/R → S
R1/R2 → R1



Here we can clearly see that right expr. is evaluated 1st, irrespective of whether operator is + or *. Hence both + & * have same precedence. And expn. is right associative.

Operator Precedence Parser (OPP)

→ OPP is simple compared to LR parser
→ OPP is less powerful compared to LR parser
→ OPP can handle ambiguous grammar also without resolving conflicts
→ OPP suitable only for operator grammar.
→ Operator grammar is a CFG in which no ∈-rules are defined and no adjacent variables on R.H.S part of grammar.

Ex:① $E \rightarrow E+E \mid E*E \mid id$

② $E \rightarrow E+T \mid T$
$T \rightarrow T*F \mid F$
$F \rightarrow id$

Not operator Grammar

① $S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b$

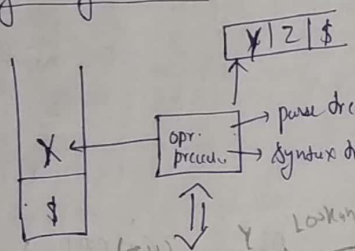
② $S \rightarrow aS \mid bS \mid \in$

OPP not suitable for these

Int
NOTE: Operator grammar allow unit productions

Scanned by CamScanner

Parsing Algorithm



T = # of terminals

Parsing table $(T+1)^2$

	id	+	*	$
id	◆	·>	·>	·>
+	<·	=	<·	·>
*	<·	·>	·>	·>
$	<·	<·	<·	accept

(T+1) — X (Stack Symbol); Y — Lookahead symbol

* ·> +
+ = +
- <· *

→ Let X is topmost terminal on the stack and Y is lookahead symbol. parser takes decision regarding S/R actions as follows:-

i) if $X <\cdot Y$ or $X = Y$ then shift Y into stack and increment lookahead symbol

ii) if $X \cdot> Y$ then there is a handle in the stack.

To perform reduce action pop symbols from stack such that terminal on the top of the stack having lesser precedence over most recently popped out terminal and push corresponding LHS non terminal into top of stack

iii) If table entry is accept then i/p string is valid

iv) If table entry is blank then there is a syntax error in the i/p.

→ Time Complexity of OPP is O(n) where n is length of the i/p"

Q How many shift actions and reduce action taken by operator precedence parser to parse the i/p "id + id * id" by using following grammar and parsing table.

E → E + E
E → E * E
E → id

shift actions = 5

reduce actions = 5

Stack	Input	Action
$ | ◆	id + id * id $	Shift
$ | id	+ id * id $	Reduce E → id
$ | E	+ id * id $	Shift
$ | E | +	id * id $	Shift
$ | E | + | id	* id $	Reduce E → id
$ | E | + | E	* id $	Shift
$ | E | + | E | *	id $	Shift
$ | E | + | E | * | id	$	Reduce E → id
$ | E | + | E | * | E	$	Reduce E → E * E
$ | E | + | E	$	Reduce E → E + E
$ | E	$	accept

Scanned by CamScanner

Construction of Operator Precedence Parsing Table

→ Size of OPP table is $(T+1)^2$ (T → no of terminals; +1 → for $).

→ To construct parsing table for unambiguous grammar Parse tree method is used.

→ To construct parsing table for ambiguous grammars general programming language precedence is are inserted in the table.

→ In the parsing table $ is lesser precedence over other operators

→ identifier (id) is higher precedence over other operators.

Ex: E → E+E | E*E | id

→ Any operator grammar contains both Left recursion and Right Recursion at one non-terminal, then that grammar is ambiguous grammar

→ Any operator grammar contains either left Recursion or Right Recursion but not both at one non-terminal. that grammar is unambiguous grammar.

Q Construct operator precedence parsing table for the following grammar and verify given grammar is operator precedence grammar or not.

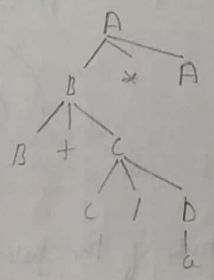
→ given grammar is unambiguous

$A \to B \times A \mid B$

$B \to B + C \mid C$ → division

$C \to C / D \mid D$

$D \to a$



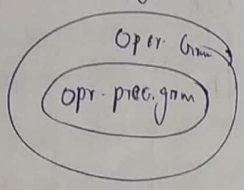
If associativity is same then check for recursion.

	a	×	+	/	$
a		·>	·>	·>	·>
×	<·	<·	<·	<·	·>
+	<·	·>	·>	<·	·>
/	<·	·>	·>	·>	·>
$	<·	<·	<·	<·	accept

NOTE: Any grammar operator precedence parsing table contains multiple entries in one box, that grammar is not operator precedence grammar. Otherwise it is operator precedence grammar.

Scanned by CamScanner

→ Every operator precedence grammar is Operator grammar, but every operator grammar need not be precedence grammar.



Q Which of the following precedence relation is true about the given grammar.

$A \to B \oplus A / B$
$B \to B \& C / C$
$C \to D @ C / D$
$D \to a$

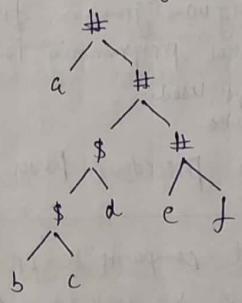
a) & and @ are left associative
b) $\oplus$ is higher precedence than @.
c) & is higher precedence than @
d) @ and $\oplus$ are right associative.

Q $X \to X \oplus Y / Y$
$Y \to Z * Y / Z$
$Z \to a$

Which of the following is true ?

a) $\oplus$ and * are left associative.
b) $\oplus$ and * are left associative
c) $\oplus$ is left associative and * is right associative
d) None of these

Gate 2018
Q Consider the following tree structure



& $ are operators and

=> $ > #
=> # is right associative
=> $ left associative

Which of the following is true

a) # is higher precedence than # $
b) $ and # are left associative
c) $ is right associative and # is left associative
d) $ is left associative and # is right associative

Scanned by CamScanner

Q Construct operator precedence parsing table for the following ambiguous grammar

$A \rightarrow A+A \mid A*A \mid a$

	a	+	*	$
a		·>	·>	·>
+	<·	·>	<·	·>
*	<·	·>	·>	·>
$	<·	<·	<·	Accept

Concept: In case of ambiguous grammar, fill table based on general programming language precedence rules is used.

Drawback of Operator precedence Parser (OPP)

→ OPP is less powerful compared to LR parser because this method will not work for ε productions and productions having adjacent variables on the R.H.S

→ Hence LR parser is powerful in bottom up Approach

Top Down Parsing (TDP)

→ TDP Top Down parser constructs parse tree, starts from the starting symbol of the grammar and proceeds towards given string by replacing non-terminal by its R.H.S part.

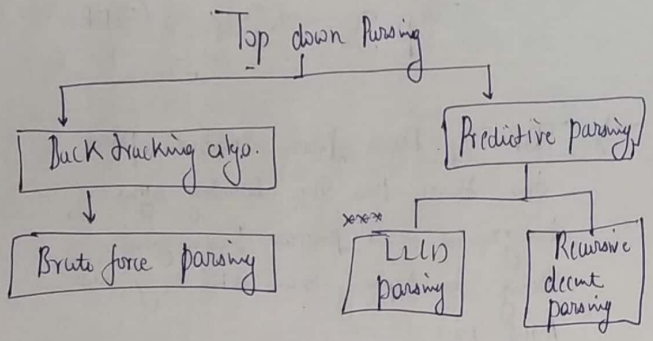
→ TDP down parsers uses Left Most Derivation while constructing parse tree

→ Difficulty in Top Dow Parsing is if a non terminals is having multiple productions then selecting correct production for tree construction is difficult.

→ There are 2 types of top down parsing algorithms exists known as
  i) Back tracking algorithm
  ii) Predictive parsing algorithm

Scanned by CamScanner

Top down Parsing



Brute force Parsing

→ Time Complexity of Brute force parsing algorithm is $O(2^n)$

→ Hence brute force parsing algorithms not suitable for practical compilers

→ In brute force parsing if a non-terminal having multiple productions then priority is given for 1st production to construct the parse tree. If the required string is not generated then parser back tracks and priority is given for 2nd production then for 3rd production and so on....

Predictive Parsing Algorithm (PPA)

→ PPA are without having back tracking

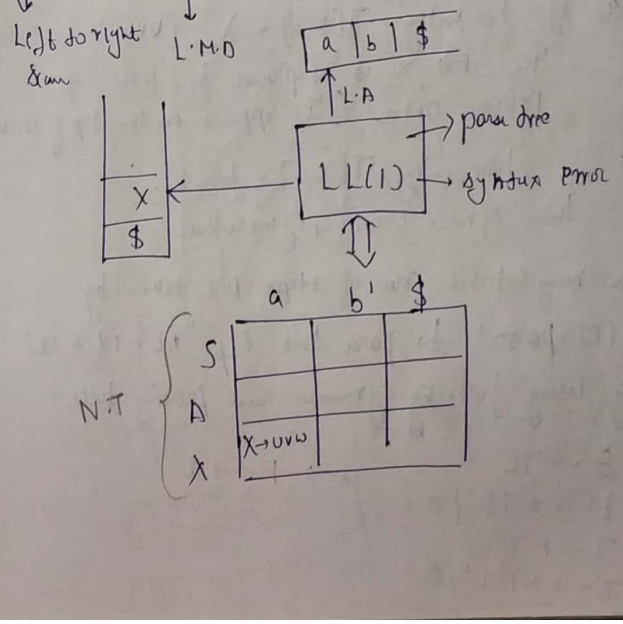
→ There are 2 types of predictive parsers known as LL(1) parser and Recursive decent parser

LL(1) parsing Algorithm

L L (1) → lookahead symbol of length 1

Left to right scan

L.M.D



Scanned by CamScanner

→ Let X is Symbol on the top of the stack and "a" is lookahead symbol, then parser takes decision as follows

① if $X = a = \$$ then i/p string is valid

② if $X = a \neq \$$ then pop X from stack and increment lookahead Symbol

③ if X is Non Terminal (N.T) in the stack then it is replaced by its R.H.S part by selecting the production from parsing table

④ if table entry $T[X,a] = X \rightarrow UVW$ then row X is replaced by "UVW" in reverse order ("U" appears on the top of stack)

⑤ if table entry $T[X,a]$ = blank, then syntax error is generated.

Q How many total no. of steps are taken by LL(1) parser to parse the i/p "id + id * id" by using following grammar and parsing table

$E \rightarrow TE'$
$E' \rightarrow +TE' / \epsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' / \epsilon$
$F \rightarrow id$

	id	+	*	$
E	$E \rightarrow TE'$			
E'		$E' \rightarrow +TE'$		$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$	$T' \rightarrow \epsilon$
F	$F \rightarrow id$			

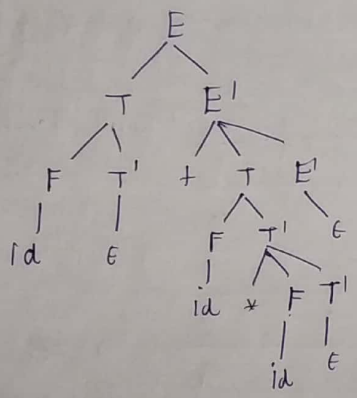
pop E and replace by its RHS part in reverse order

Stack	Input	Operation
$E	id + id * id $	$E \rightarrow TE'$
$E'T	id + id * id $	$T \rightarrow FT'$
$E'T'F	id + id * id $	$F \rightarrow id$
$E'T'id	id + id * id $	pop
$E'T'	+ id * id $	$T' \rightarrow \epsilon$
$E'	+ id * id $	$E' \rightarrow +TE'$
$E'T+	+ id * id $	pop
$E'T	id * id $	$T \rightarrow FT'$
$E'T'F	id * id $	$F \rightarrow id$
$E'T'id	id * id $	pop
$E'T'	* id $	$T' \rightarrow *FT'$
$E'T'F*	* id $	pop
$E'T'F	id $	$F \rightarrow id$
$E'T'id	id $	pop
$E'T'	$	$T' \rightarrow \epsilon$
$E'	$	$E' \rightarrow \epsilon$

Scanned by CamScanner

Stack	Input	Operation
$ E' T	id $	T → F T'
$ E' T' F	id $	F → id
$ E' T'	$	T → ε
$ E'	$	E' → ε
$	$	accept

$T(n) \rightarrow O(n)$



17/10/18

Construction of LL(1) parsing Table

Q Construct LL(1) parsing table for the following grammar.

Q Verify given grammar is LL(1) grammar or not

$S \rightarrow aABC$
$A \rightarrow a/bb$
$B \rightarrow a/\epsilon$
$C \rightarrow b/\epsilon$

NOTE:

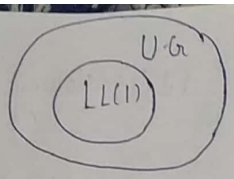
→ To construct LL(1) parsing table, ~~rows~~ : row of table decided based on L.H.S non-terminal and column decided based on R.H.S part first set information

→ If grammar R.H.S first set contains "ε" then "follow" of non-terminal is used to decide about columns

→ Any grammar LL(1) parsing table contains multiple productions in the same box, that grammar is not LL(1) grammar

→ Every LL(1) grammar is unambiguous grammar but every unambiguous grammar need not be LL(1) grammar

Scanned by CamScanner



U·Gr
Unambiguous grammar

→ Any ambiguous grammar LL(1) parsing table should contain multiple entries.

$S \to aABC$
$A \to a \mid bb$
$B \to a \mid \epsilon$
$C \to b \mid \epsilon$

	a	b	$
S	S→aABC		
A	A→a	A→bb	
B	B→a	B→ϵ	B→ϵ
C		C→b	C→ϵ

No conflict Hence given grammar is LL(1)
multiple entries in single cell

② $S \to aAbB \mid bAaB \mid \epsilon$
$A \to S$
$B \to S$

	a	b	$
S	S→aAbB (S→ϵ)	S→bAaB (S→ϵ)	S→ϵ
A	A→S	A→S	
B	B→S	B→S	B→S

Hence given Gr is not LL(1)

③ $S \to aSA$
$A \to c \mid \epsilon$

Given Gr is not LL(1)

	a	c	$
S	S→aSA		
A		A→c A→ϵ	A→ϵ

④ $S \to iEtSS'$
$S' \to eS \mid \epsilon$
$E \to b$

$F(S') = \{\$, e\}$

	i	e	t	b	$
S	S→iEtSS'				
S'		S'→eS S'→ϵ			S'→ϵ
E				E→b	

Hence Gr is not LL(1)

LL(1) Grammar Detection

① Single Production grammar is always LL(1) grammar.

② Any grammar is of the form
$A \to \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid \dots \mid \alpha_n$
then $First(\alpha_1), First(\alpha_2), \dots, First(\alpha_n)$ are pair wise disjoint. then grammar is LL(1)
(No common element exists)

③ If the grammar is of the form
$A \to \alpha \mid \epsilon$
then $First(\alpha) \cap Follow(A) = \phi$ then grammar is LL(1) otherwise not LL(1)

④ Ambiguous grammar is not LL(1) grammar

Q Which of the following grammar is LL(1) grammar

		First	First	Follow
(a)	$S \to aSbS \mid bSaS \mid \epsilon$ ✗	{a}	{b}	{b, a}
(b)	$S \to aSa \mid bSb \mid \epsilon$ ✗	{a}	{b}	{a, b}
(c)	$S \to AaAb \mid BbBa$ ✓ $A \to \epsilon$ $B \to \epsilon$	{a}	{b}	?
(d)	None			

Scanned by CamScanner

Q Which of the following grammar is unambiguous grammar (check for LL(1))

(a) $S \to A/a$ ✗ First {a} First {a}
$A \to a$

(b) $S \to AA$ ✗ First {a} First {a}
$A \to aA/a$

(c) $S \to aS/bSb/c$ {a} {bb} {c}

(d) None

NOTE: Left recursive grammars are not LL(1) grammars but elimination of left recursion may results LL(1).

Ex: $S \to Sa/b$ ← not LL(1)

$S \to bS'$
$S' \to aS'/\epsilon$ } given is LL(1)

① Any Non-deterministic grammar is not LL(1) grammar but Left factoring of that grammar may results into LL(1) grammar.

Non-Deterministic grammar

In any grammar if a non-terminal having multiple productions then common prefix exists in the R.H.S part productions.

Ex: $S \to ab/ac$
$S \to iEtS/iEtSeS/a$

Q Left factor following CFG

① $S \to ab/ac$

$S \to aS'$
$S' \to b/c$

② $S \to iEtS/iEtSeS/a$
$E \to b$

$S \to iEtSS'/a$
$S' \to \epsilon/eS$
$E \to b$

Gate

Q Which of the following is sufficient to convert CFG into LL(1) grammars

(a) Elimination of Left Recursion only

(b) Left factoring of grammar only

(c) Both (a) & (b)

(d) None of these

NOTE: Converting CFG into LL(1) grammar is undecidable problem.

CFG represents CFL's and LL(1) grammar represents DCFL, Hence converting CFG into LL(1) grammar means converting every CFL into DCFL which is impossible.

Scanned by CamScanner

Q Which of the following is True about the given grammar

$S \to Aca \mid Bcb$ ~~$Aca \mid Bcb$~~
$A \to c$
$B \to c$

① LL(1) grammar
② LL(2) grammar
③ both LL(1) & LL(2)
④ neither LL(1) nor LL(2)

Sol: Concept:- LL(2) → Lookahead Symbol of length 2

In LL(2) we have to look for 2 symbols.

$S \to Aca \mid Bcb$
$cca \mid ccb$
→ both are same (ie common element)

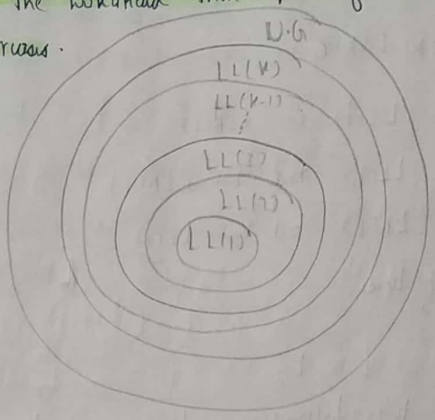
Hence G is not LL(2)

First(Aca) ∩ First(Bcb) ≠ ϕ

NOTE: Every LL(K-1) grammar is LL(K) grammar but every LL(K) need not be ~~LL(K)~~ LL(K-1).

→ Every LL(K) grammar should be unambiguous grammar

→ In top down parsing if we increase length of the lookahead then power of the parser also increases.



NOTE:

① In bottom up parsing LR(K) is equivalent to LR(1) LR(1) = LR(2) = LR(3) = ... LR(K)

② In bottom up parsing if we ~~increase~~ increase length of the lookahead, power of the parser not increases.

③ In bottom up parsing power of the parser depends on way of placing reduce actions in parsing table.

Scanned by CamScanner

Q. Which of the following is true about given grammar.

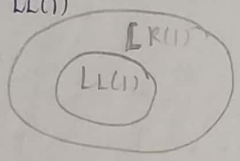
$S \rightarrow aSa \mid bSb \mid c$

(a) It is LL(1) & LR(1)

(b) It is LL(1) but not LR(1)

(c) It is LR(1) but not LL(1)

(d) None of these.



Q. Which of the following is true about the given grammar

~~S → AB~~ $E \rightarrow E+n \mid n$

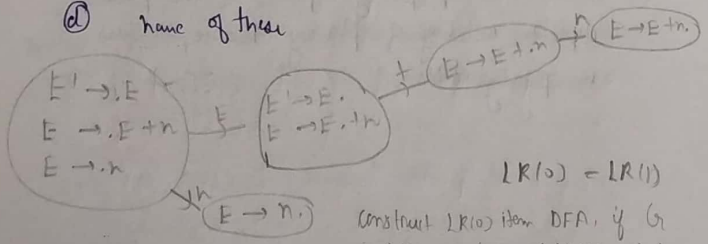
~~A → a~~

(a) It is LL(1) & LR(1)

(b) It is LR(1) but not LL(1)

(c) neither LL(1) nor LR(1)

(d) None of these



LR(0) = LR(1)

Construct LR(0) item DFA, if Gr is LR(0) then definitely it is LR(1)

NOTE:

→ Left Recursive grammars and non-deterministic grammars can be parsed by bottom up parser but not by Top down parser

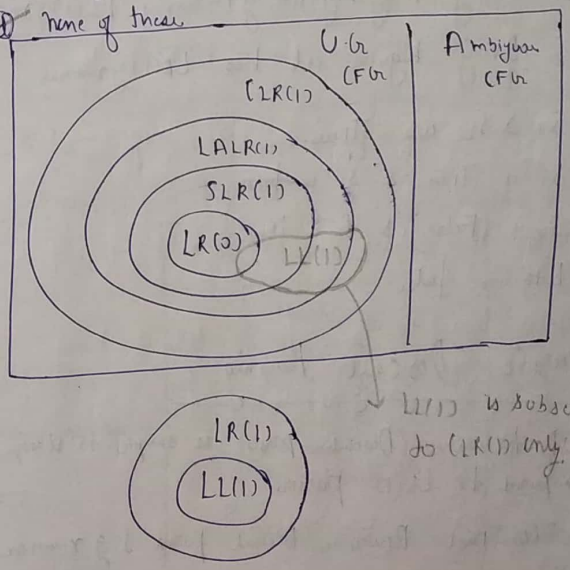
Q. Which of the following is True

(a) Every LL(1) grammar is LR(0) grammar

(b) Every LL(1) grammar is SLR(1) grammar

(c) Every LL(1) grammar is ~~LR(1) gr~~ LALR(1) grammar

(d) None of these.



LL(1) is subset to LR(1) only.

Scanned by CamScanner

NOTE:

→ For every DCFL, LR(1) should exists but LL(k) may or may not exists.

→ Every Regular language DCFL, hence for every R-L also LR(1) grammar should exists but LL(k) may or may not exists.

Q Consider following 2 statements

S_1: Every Regular grammar is LL(1) grammar

S_2: Every Regular set has LR(1) grammar

ⓐ S_1 & S_2 are True.

ⓑ S_1 is True & S_2 is false.

ⓒ S_1 is False & S_2 is true

ⓓ Both are false

Recursive Descent Parsing

→ Recursive Descent parser is simple to design compared to LL(1) parsers.

→ To construct Recursive Descent parser, grammar should not contain left recursion. And it should be left factored grammar.

→ In Recursive descent parser corresponding to every non-terminal one procedure is constructed, these procedures may be recursive procedures.

Draw back

Recursive descent parser suitable only for small class of grammars. Hence power of this parser is less compared to LL(1) parser.

Q Construct Recursive Descent Parser for the following grammar

$E \rightarrow num\ T$

$T \rightarrow * num\ T \mid \epsilon$

```
S.T:-  main()
       {
          E()
       }

match()
{
   lookahead = next token;
}

error()
{
   print("Syntax Error")
}

Proc. E()
{
   if (lookahead == num)
   {
      Match();
      T();
   }
   else { error(); }
   if (lookahead == $)
      print("Success");
   else
      error();
}

Proc. T()
{
   if (lookahead == *)
   {
      match();
      if (lookahead == num)
      {
         match();
         T();
      }
      else { error(); }
   }
   else
      return null;
}
```

Scanned by CamScanner

Drawback of Parser

→ Parser can detect syntax errors only and can not detect semantics errors

→ Following are some of the semantic errors parser can not detects

1. Using a variable without declaration.
2. multiple times declaration of same variable in the same scope.
3. Expression statements which are not type compatible.
4. Miss match of formal and actual parameters. in functions
5. Type miss matches in formal and actual parameter

Q Which of the following C- prog. statements having syntax errors

(a) If (z); id(z)
(b) For (a,b,c); id(a,b,c)
(c) While (a,b); id(a,b)
(d) none

Q Which of the following C- prog. statements having syntax errors

(a) for (i=; i<10; i++)
(b) int a,b,a,a;
(c) if ( ); → condition expr. must be present
(d) none

Q Which of the following C- prog. having syntax error

(a)
```
main()
{
  int a,b;
  c=a+b;
}
```

(b)
```
main()
{
  int a,b,a,b;
}
```

(c)
```
main()
{
  int a,b;
  char c[] = "Compiler";
  a=b+c;
}
```

(d) none.

Scanned by CamScanner

Semantic Analysis Phase

→ To design Semantic analysis phase CSG's are used. Hence Semantic analysis phase is also called as Context-Sensitive Analysis.

→ CSG's are difficult to implement, hence "Attribute grammars" are used to Design the Semantic analysis phase.

→ Attribute grammar is a CFG where each grammar symbol is associated with attribute information

→ Attribute can be a type of a variable or scope of a variable or it can be intermediate code also.

Syntax Directed Translation (SDT)

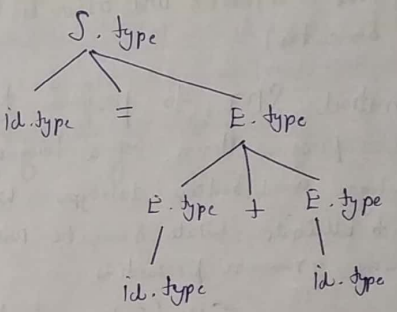
SDT is a notation where each CFG production is associated with set of semantic rules.

SDT = CFG + { Semantic rules }

→ Semantic rules are, rules to compute attribute information in Annotated Parse tree.

Annotated Parse Tree (or) Decorated Parse Tree

It is a parse tree where each grammar symbol is decorated with attributes



Applications of SDT :-

① SDT is used to detect Semantic errors.
② SDT is used to construct Symbol table
③ SDT is used to generate intermediate code
④ SDT is used to design DAG.
⑤ Constructed SDT are given to TDP or BUP.

Scanned by CamScanner

⑥ Whenever parser performs parsing with grammar productions, then corresponding Semantic rules are also executed which is associated with grammar production

⑦ The parser used in Semantic analysis contains 2 stacks, one is used for grammar symbols (Syntax) and other is used for attributes (Semantics)

Q Construct SDT to perform type checking on expression stmts of a language having integer and boolean datatypes. where type is a ↳ attribute. Write Semantic rules for the following grammar productions

$E \rightarrow E_1 + E_2 \rightarrow$ { if ($E_1.type == E_2.type$) && ($E_1.type = int$) then $E.type = int$ else error() }

$E \rightarrow (E_1)$

$E \rightarrow E_1 == E_2$ { if ($E_1.type == E_2.type$) then $E.type = bool$ else error }

~~$E \rightarrow$ Num or Num~~

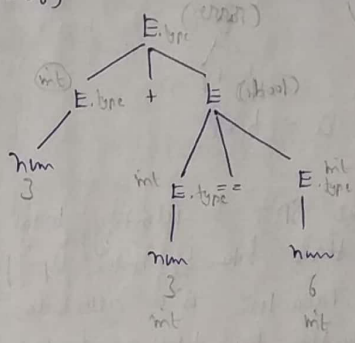
$E \rightarrow num$ { $E.type = int$ }

$E \rightarrow True$ { $E.type = bool$ }

$E \rightarrow false$ { $E.type = bool$ }

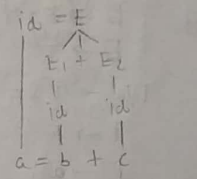
$E \rightarrow (E_1)$ { $E.type = E_1.type$ }

$3 + (3 == 6)$



SDT for detection of ~~incompatible~~ type incompatible Expressions

```
main()
{
  int a,b
  char c[] = "Compiler";
  a = b+c;
}
```

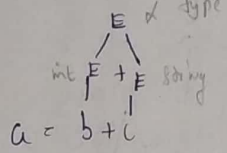


$S \rightarrow id = E$

$E \rightarrow E_1 + E_2$ { if ($E_1.type == E_2.type$) && ($E_1.type = int$) then $E.type = int$ else error }

$E \rightarrow id$ { $E.type = id.type$ }
↳ get this from Symbol table.

Scanned by CamScanner



type miss match
incompatible data types
string

Q Construct SDT to identify total no. of reductions taken by bottom up parser where reduction is a attribute. Root value E.reduction use total no of reductions value. Write semantic rules by using following grammar productions.

$E \rightarrow E+T$
$E \rightarrow T$
$T \rightarrow T*F$
$T \rightarrow F$
$F \rightarrow a$

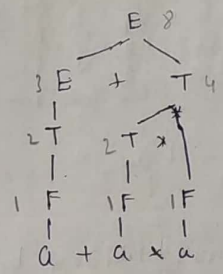
Steps to Construct SDT's

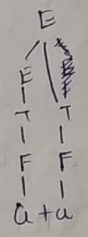
① Construct Parse tree for some i/p which includes all grammar productions

② Construct Annotated Parse tree.

③ Compute attribute information in the annotated parse tree.

④ Generalize the rules of attribute computation in the annotated parsetree, attach to grammar productions as semantic rules.





$E \rightarrow E_1 + T$ { E.red = E_1.red + T.red + 1 }

$E \rightarrow T$ { E.red = T.red + 1 }

$T \rightarrow T_1 * F$ { T.red = T_1.red + F.red + 1 }

$T \rightarrow F$ { T.red = F.red + 1 }

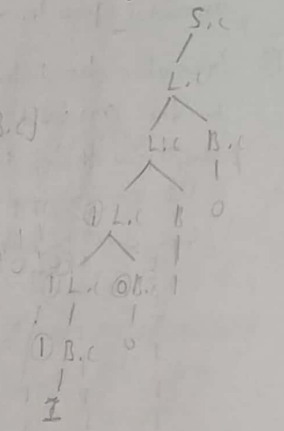
$F \rightarrow a$ { F.red = 1 }

Q Construct SDT that counts total no of 1's present in the given binary string. Where count is a attribute & root value S.count gives total no of 1's information.

Scanned by CamScanner

Write Semantic rules for the following grammar productions

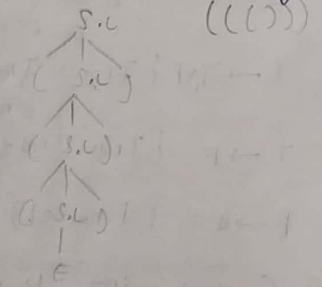
$S \to L$ { S.c = L.c }
$L \to LB$ { L.c = L.c + B.c }
$L \to B$ { L.c = B.c }
$B \to 0$ { B.c = 0 }
$B \to 1$ { B.c = 1 }



Q Construct SDT that gives total no of balanced parenthesis where count is attribute. Root value S.count give total no. of balanced parenthesis value. Write Semantic rules for the for the following grammar productions

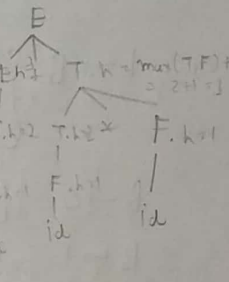
(((())))

S → (S)
S → ϵ



$S \to (S)$ { S.count = S_1.count + 1 }
$S \to \epsilon$ { S.count = 0 }

Q Construct SDT to compute height of the parse tree where height is the a attribute. final root value E.height = max total height of the tree. Write Semantic rules for the following grammar productions



$E \to E + T$ { E.h = max(E.h, T.h) + 1 }
$E \to T$ { E.h = T.h + 1 }
$T \to T * F$ { T.h = max(T.h, F.h) + 1 }
$T \to F$ { T.h = F.h + 1 }
$F \to id$ { F.h = 1 }

Q Construct the SDT that performs type conversion in Expression Statements of a language having integer and real types. where type is a attribute. Write Semantic rules for the following grammar productions.

$E \to E + T$ { if (E.type == T.type) && (E.type = int) then E.type = int else E.type = real }
$E \to T$ { E.type = T.type }
$T \to num$ { T.type = int }
$T \to num.num$ { T.type = real }

Scanned by CamScanner

Q. Consider the following SDT

$F \to L \quad \{ F.v = L.v \}$

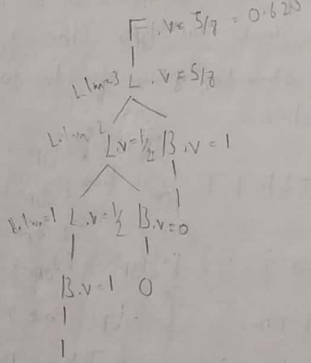
$L \to L_1 B \quad \left\{ \begin{array}{l} L.l_m = L_1.l_m + 1 \\ L.v = L_1.v + \frac{B.v}{2^{L.l_m}} \end{array} \right\}$

$L \to B \quad \left\{ \begin{array}{l} L.l_m = 1 \\ L.v = \frac{B.v}{2} \end{array} \right\}$

$B \to 0 \quad \{ B.v = 0 \}$

$B \to 1 \quad \{ B.v = 1 \}$

Find $F.v = ?$ for i/p = 101.



Q Consider the following SDT & Compute E.val for i/p : $ 2 # 3 & 5 # 6 & 4

$E \to E_1 \# T \quad \{ E.val = E_1.val * T.val \}$

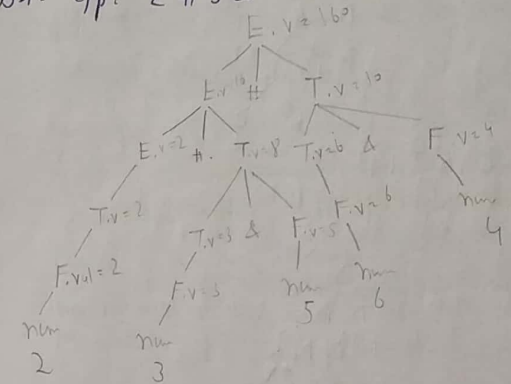
$E \to T \quad \{ E.val = T.val \}$

$T \to T_1 \& F \quad \{ T.val = T_1.val + F.val \}$

$T \to F \quad \{ T.val = F.val \}$

$F \to num \quad \{ F.val = num.val \}$

Sol: i/p: 2 # 3 & 5 # 6 & 4



Q. Consider the following SDT

$S \to AS \quad \{ print(1) \}$

$S \to AB \quad \{ print(2) \}$

$A \to a \quad \{ print(3) \}$

$B \to bC \quad \{ print(4) \}$

$B \to dB \quad \{ print(5) \}$

$C \to c \quad \{ print(6) \}$

a) This SDT is carried out along with BUP. What o/p will it produce for i/p "aadbc"

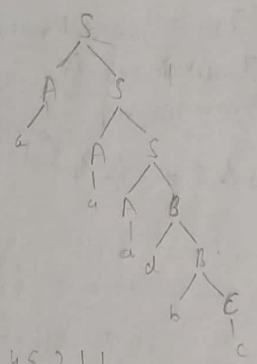
Scanned by CamScanner

b) This SDT is carried out along with TDP. What o/p will it produce for i/p "aaadbc".

aaadbc

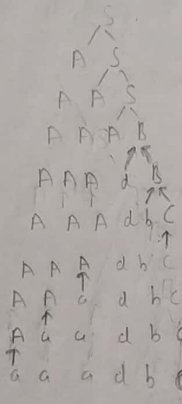
Concept: Scan from top to down in post order produce o/p

Sol: b)



o/p → 3 3 3 6 4 5 2 1 1

a)



3 3 3 6 4 5 2 1 1

NOTE: If SDT is carried out along with TDP (or) BUP o/p is same for the given same i/p

② If SDT is carried out along with BUP, corresponding to every reduce action the semantic rule associated with that production will be executed.

③ If the SDT is carried out along with TDP, then semantic rules are executed in depth first search left to right manner in which if a non-terminal visited 2nd time the semantic rule corresponding to that non terminal is executed

Q Consider the following SDT.

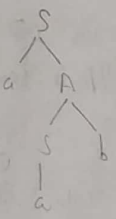
S → aA { print (1) }
S → a { print (2) }
A → Sb { print (3) }

This SDT is carried out along with shift reduce parser. Then what o/p will it produce for the i/p "aab".

Concept: Shift reduce means BUP. We get same o/p if we generate it through TDP.

Scanned by CamScanner

Sol:-



o/p → 2 3 1

Synthesized Attribute

→ If the attribute value at a node in annotated parse tree computed from children only, known as synthesized attribute

→ Synthesized attributes are ~~well~~ very well suited for Bottom Up Parser. ~~(BU)~~

Inherited Attributed

→ If the attribute value at a node in annotated parse tree computed from parent or from left sibling known as inherited attribute.

→ Inherited attributes are very well suitable for Top down parser.

NOTE:

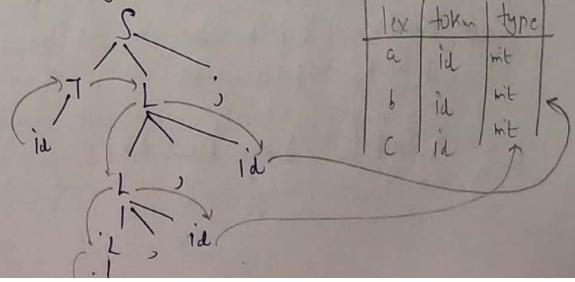
→ Lexical analyzer compt constructs symbol table partially but not completely.

→ Complete symbol table construction is done at semantic analysis phase by using SDT.

→ Symbol table construction SDT uses synthesized attributes and inherited attributes.

Q ~~Consider~~ ~~Construct~~ (construct) the SDT to construct complete symbol table by inserting type information of ~~various~~ variables where "S.type" is synthesized attribute and "itype" is inherited attribute. Write semantic rules by using following grammar productions.

S → TL { L.itype = T.stype }
T → int { L.stype = int }
T → float { L.stype = float }
L → L1, id { L1.itype = L.itype, addtype("id.name", L.itype) }
L → id { addtype("id.name", L.itype) }



lex	token	type
a	id	int
b	id	int
c	id	int

Scanned by CamScanner

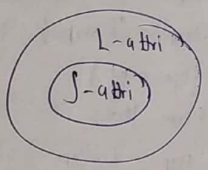
S- attributed Definition

SDT that contains only synthesized attributes known as S- attributed definition

L- attributed Definition

SDT that contains both synthesized attributes and inherited attributes known as L-attributed definition.

→ Every S- attributed definition is L- attributed definition but every L-attributed need not be S-attributed.



Ex: S → AB { B.x = A.x }
A → a { A.x = a }
B → b { B.x = b }

Ex: S → AB { S.x = A.x + B.x }

Translation Scheme (SDT)

→ Translation scheme is SDT in which semantic rules may exists in b/w grammar symbols

→ In translation scheme all semantic rules corresponding to synthesized attributes present at Right most end of grammar production & all inherited attributes of a non-terminal, semantic rule present before that non-terminal

Ex: S → T { L.itype = T.stype } L ;
T → int { T.stype = int }
T → float { T.stype = float }
L → { L1.itype = L.itype } L1 , id { addtype ("id.name", L.itype) }
L → id { addtype ("id.name", L.itype) }

NOTE:

→ In translation scheme we can execute semantic rules in depth first search left to right manner.

→ Whenever parse tree is constructed from translation scheme, an edge should be provided for semantic rule also. (Semantic rule is treated as one of the grammar symbol)

Scanned by CamScanner

Great

Q Consider the following translation scheme and what o/p it will produce for the i/p 9+5+2

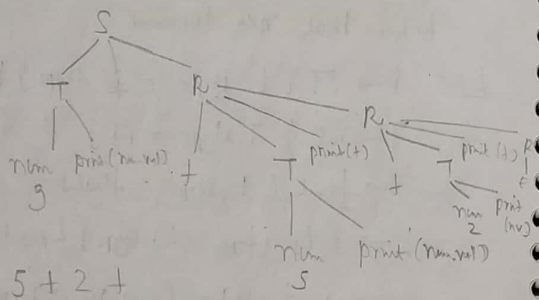
$S \rightarrow T R$

$R \rightarrow + T \{ print(+) \} R$

$R \rightarrow \epsilon$

$T \rightarrow num \{ print(num.val) \}$

Sol:



o/p → 9 5 + 2 +

Q Consider the following translation scheme, and what o/p it will produce for the i/p 2*3+4

$S \rightarrow E R$

$R \rightarrow * E \{ print(*) \} R$

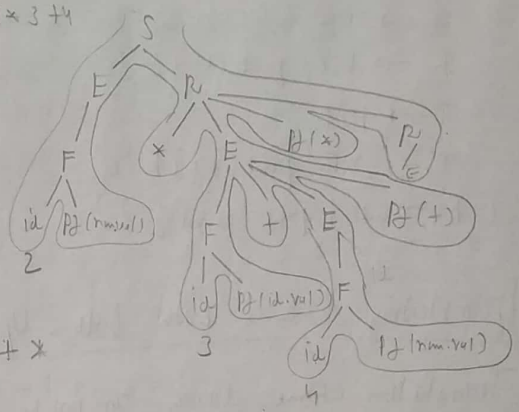
$R \rightarrow E$

$E \rightarrow F + E \{ print(+) \}$

$E \rightarrow F$

$F \rightarrow id \{ print(id.val) \}$

Sol: 2*3+4



o/p → 2 3 4 + *

Translation Scheme with Top Down Parser

→ if the translation scheme is carried out with TDP, then Left recursion should be eliminated from it.

→ Way of elimination of Left recursion is same as elimination of Left recursion from grammar.

Q Eliminate Left recursion from the following translation scheme

$E \rightarrow E + T \{ print(+) \}$

$E \rightarrow T$

$T \rightarrow T * F \{ print(*) \}$

$T \rightarrow F$

$F \rightarrow id \{ print(id.val) \}$

Scanned by CamScanner

Sol:- $E \rightarrow TE'$

$E' \rightarrow +T\{print(+)\}E'/\epsilon$

$T \rightarrow FT'$

$T' \rightarrow *F\{print(*)\}T'/\epsilon$

$F \rightarrow id\ \{print(id.val)\}$

Translation Scheme with Bottom Up Parser

→ Translation scheme directly can not be parsed by BUP. bcz handle can not be detected due to middle semantic rule, Hence translation scheme is modified with the help of new non-terminals by taking all semantic rules at right most end of grammar productions

→ Corresponding to every reduce action the semantic rule ~~rules~~ associated with that ~~new~~ non-terminal is executed.

Q Convert the following translation scheme suitable for Bottom up Parser?

$E \rightarrow T + E'$

$E' \rightarrow +T\{print(+)\}E'/\epsilon$

$T \rightarrow FT'$

$T' \rightarrow *F\{print(*)\}T'/\epsilon$

$F \rightarrow id\ \{print(id.val)\}$

Sol:- $E \rightarrow TE'$

$E' \rightarrow +TME'/\epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FNT'/\epsilon$

$F \rightarrow id\ \{print(id.val)\}$

$M \rightarrow \epsilon\ \{print(+)\}$

$N \rightarrow \epsilon\ \{print(*)\}$

Drawback of SDT

By using SDT we can detect static semantic errors only but not dynamic semantic errors.

Statically typed language

→ The language in which type checking is done at compile time only, by using SDT.

Dyn

Scanned by CamScanner

Dynamically typed language

→ The lang. in which type checking is done at run time by using additional code.

→ SDT can not detect dynamic semantic errors

→ Ruby

Untyped languages

The lang. in which type checking is done. (no)

Ex:- Assembly language

Q. Which of the following is false.

(a) In statically typed languages each prg. variable has fixed type

(b) In Untyped languages values do not have any types.

(c) In dynamically typed languages program variables do not have any types.

(d) none of these

18/10/18

Intermediate Code Generation Phase

Intermediate code is a type of code that is easily generated from High level language and easily converted into low level language.

If intermediate code is generated then performing m/c independent optimization is possible.

following are different type of intermediate code used in compiler

Intermediate code generation

Syntax Tree

→ Syntax tree is optimized form of parse tree

→ Syntax tree represents order of evaluation of expression statements in proper precedence and associativity rules.

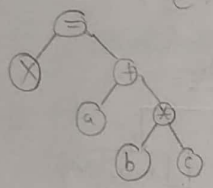
Scanned by CamScanner

→ In Syntax tree common subexpression are repeatedly evaluated Ex. X = b*c + b*c

→ In syntax tree, corresponding to every subexpression, operands present at leaf position and Operator present at root position

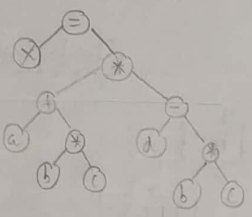
Q Construct syntax tree for the following expression statements and identify how many edges present in it.

① X = a + b*c



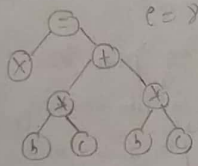
e = 5

② X = (a+b*c) * (b*c-d)



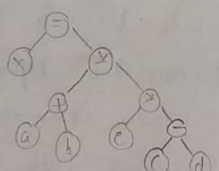
e = 12

③ X = b*c + b*c



e = 9

④ X = (a+b) * ((e*(c-d)))

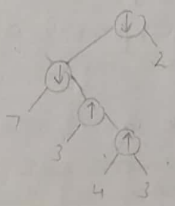


e = 10

Q How many edges present in syntax tree corresponding to following expression 7 ↓ 3 ↑ 4 ↑ 3 ↓ 2

precedence of ↑ is higher than operator ↓ and ↑ is right associative & ↓ is left associative

Sol ((7 ↓ (3 ↑ (4 ↑ 3)) ↓ 2)



Drawback of Syntax Tree

In syntax tree common subexpression are repeatedly evaluated, hence time & space required in the target code is more compared to DAG representation.

Directed Acyclic Graph (DAG)

→ DAG is a syntax tree in which common subexpressions are eliminated

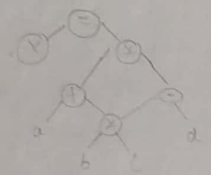
→ Advantage of DAG representation is some type of optimization automatically performed.

Scanned by CamScanner

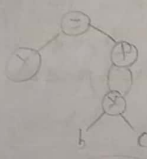
→ For any expression if DAG is constructed then common sub expression elimination optimization is automatically performed

Q How many edges present in the DAG for the following expression statements

① $X = (a+b*c) * (b*(-d))$

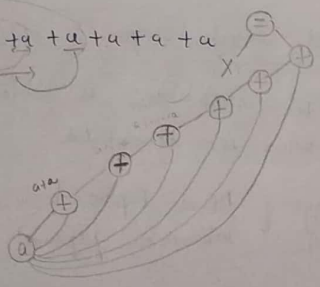


② ~~$X = (b*c) + b = X = b*c + b*c$~~

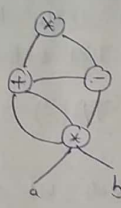


Imp (b)

$X = a+a+a+a+a+a+a+a$



Q (Imp) Construct expression ~~ex~~ equivalent to

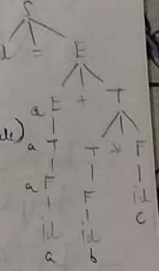


$((a*b) + (a*b)) * (((a*b)+(a*b)) - (a*b))$

NOTE: Tree form intermediate code is not suitable to represent flow control statements and functions Hence ~~Tree form of~~ Three address form of intermediate code is used

Q ~~Generate~~ Construct SDT to construct syntax trees by using following grammar productions.

$S \rightarrow id = E$ {S.node = makenode(id.name, '=', E.node)}
$E \rightarrow E + T$ {E.node = makenode(E.node, +, T.node)}
$E \rightarrow T$ (E.node = T.node)
$T \rightarrow T * F$ {T.node = makenode(T.node, '*', F.node)}
$T \rightarrow F$ {T.node = F.node}
$F \rightarrow id$ {F.node = id.name}



NOTE:

Scanned by CamScanner

Note:- Here node is a attribute that gives syntax tree at every grammar symbol and ~~make node~~

makenode() ~~makenode()~~ is a fun that creates root node and assigns left subtree and right subtree

Three Address Code (~~TAC~~) (TAC)

~~TAC~~ TAC is a type of intermediate code where each high level stmnt is represented by using at most 3 addresses

Follow are different forms of 3 address code :-

(Expressions)
1. $X = Y \text{ op } Z$
2. $X = \text{op } Y$
3. $X = Y$

(arrays)
4. $X = Y[i]$
5. $X[i] = Y$

(flow control)
6. goto L.
7. if a rb goto L_p
8. goto L_2

(function)

Param P_1
Param P_2
⋮
Param P_n
Call name, arguments

→ While generating 3 address code compiler generates temporary variables

→ In any subexpression if 2 addresses are filled compiler generates temporary variable

Gate

Q How many temporary variables generated by the compiler for the following expression's stmnts in 3 address code?

① $X = a + b * c$

#temp = 2

$t_1 = b * c$
$t_2 = a + t_1$
$X = t_2$

② $X = (a+b) * (e + (c-d))$

#temp = 4

$t_1 = a + b$
$t_2 = c - d$
$t_3 = e + t_2$
$t_4 = t_1 * t_3$
$X = t_4$

③ $X = a*b + (c*d - e*f)$

#temp = 5

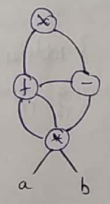
$t_1 = a*b$
$t_2 = c*d$
$t_3 = e*f$
$t_4 = t_1 + t_2$
$t_5 = t_4 - t_3$
$X = t_5$

④ $X = (a*b) + (c*d)/(a+b)$

#temp = 5

$t_1 = a*b$ / $t_1 = a*b$
$t_2 = c*d$ / $t_2 = a+b$
$t_3 = t_2 / t_1$ / $t_3 = c*d$
$t_4 = t_1 + t_3$ / $t_4 = t_3 / t_1$
$X = t_4$ / $t_5 = t_1 + t_4$
$X = t_5$

⑤



$t_1 = a * b$
$t_2 = t_1 + t_1$
$t_3 = t_2 - t_1$
$t_4 = t_2 * t_3$
$X = t_4$

NOTE: 3 address code for expression is also known as Static Single assignment (SSA)

→ In SSA form every expression is assigned to new variable, hence reusing ~~using~~ of ~~vars~~ variables is not allowed in SSA form.

Scanned by CamScanner

Ques

Q Which of the following Sato state single assignment is equivalent to following 3 address code address instruction.

```
p = a-b
q = p*c
p = u*v
q = p+q
```

(a)
```
P1 = a-b
q1 = P1 * C
P1 = U * V
q1 = P1 + q1
```

(b)
```
P1 = a-b
q1 = P1 * C
P3 = U * V
P2 = P1 + q3
```

(c)
```
P3 = a-b
q1 = P3 * c
P4 = U * V
q5 = P4 + q4
```

(d) none

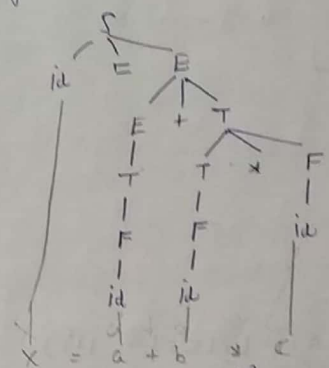
Q Construct the SDT to generate 3 address code of expression & statements by using following grammar productions

S → id = E
E → E + T
E → T
T → T * F
T → F
F → id

Here "new temp()" is a fun" that creates a new temporary variable and "place" is a attribute that gives intermediate code at every place. Write semantic rules for the given ~~Grammar~~ grammar productions

Sol:- given grammar generates such kinds of productions

X = a + b * c



S → id = E { gencode (id.name = E.place) }
E → E + T { t = newtemp(); gencode (t = E.place + T.place); E.place = t }
E → T { E.place = T.place }
T → T * F { t = newtemp(); gencode (t = T.place * F.place); T.place = t }
T → F { T.place = F.place }
F → id { F.place = id.name }

Q Construct 3 address code for the following flow control Stmts & how many temporary variables are generated by the compiler

```
for (i=1; i<10; i++)
{
    a = b + c;
}
```

Scanned by CamScanner

(i) for (i = 1; i < 10; i++)
{
a = b + c;
}

i = 1
L_0: if i < 10 goto L_1
goto L_2
L_1: t_1 = b + c
a = t_1
t_2 = i + 1
i = t_2
goto L_0
L_2: exit

(ii) for (i = 1; i <= 100; i++)
{
x = (y * z) + (a + b * c);
}

i = 1
L_0: if i <= 100 goto L_1
goto L_2
L_1: t_1 = y * z
t_2 = b * c
t_3 = a + t_2
t_4 = t_1 + t_3
x = t_4
t_5 = i + 1
i = t_5
goto L_0
L_2: exit

if-else Statement

Q Construct 3 address code for the following if-else statement

```
if ((a>5) (or) ((c<d) and (e>f)))
  {
    a = b+c;
  }
else
  {
    x = y+z;
  }
```

NOTE:
While generating 3 address code for flow control statements compiler uses 'back patching'

In Back patching initially code is generated and goto table information is filled after generation of the complete code.

L_1: if a > b goto L_7
L_2: goto L_3
L_3: if c < d goto L_5
L_4: goto L_{10}
L_5: if e > f goto L_7
L_6: goto L_{10}

L_7: t_1 = b + c
L_8: a = t_1
L_9: goto L_{12}
L_{10}: t_1 = y + z
L_{11}: x = t_1
L_{12}: exit

Scanned by CamScanner

→ Generated 3 address code is divided into basic blocks.

→ Basic block is a sequence of 3 address code instructions which is having single entry and single exit property

→ Code Optimization phase enters into every basic block and performs optimization

→ Basic blocks are detected based on Leader statements.

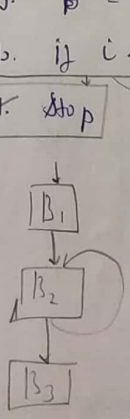
→ following are the leader statements. in 3 address code instruction

1. First statement is always a leader
2. Target of conditional ~~const~~ or unconditional code is a leader
3. The statement immediately followed by conditional (or) unconditional goto. is a leader

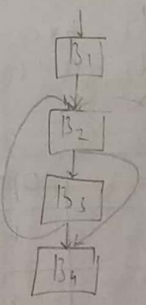
→ Within basic blocks control flow graph is constructed.

Q Convert the following 3 address code instructions into basic blocks. and construct control flow graph.

1. print f
2. i = 1
3. $t_1 = i * 4$
4. $t_2 = a[t_1]$
5. $t_3 = 4 * i$
6. $t_4 = b[t_3]$
7. $t_5 = t_2 * t_1$
8. $t_6 = p * t_5$
9. $p = t_6$
10. if $i < 20$ goto (3)
11. stop



1. $f = 1$
2. $i = 2$
3. if $i \geq n$ goto 8
4. $f = f * i$
5. $t = t + 1$
6. $i = t$
7. goto (3)
8. $i = 5 + j$
9. stop



Scanned by CamScanner

Representation of Three Address Code

→ Generated 3 address code is implemented in quadruple notation

→ Quadruple is an array having 4 fields where 2 fields are used for the operands, one for operator and other for results.

→ following is a quadruple notation for expression

$x = a + b * c$

$t_1 = b * c$

$t_2 = a + t_1$

$x = t_2$

Result	Op	Opn 1	Opn 2
t_1	*	b	c
t_2	+	a	t_1
x	=	t_2	

Drawback of quadruple

→ In quadruple notation all program variables and temporary variables are stored in result field. Hence compiler is unable to differentiate b/w program variable and temporary variable Hence space for temporary variables is allocated in the symbol & tables.

→ To avoid this drawback, "Triple" representation is used. "Triple" is nothing but removing result field from quadruple notation.

Triple

	Op	Opn 1	Opn 2
100	*	b	c
104	+	a	(100)
	=	x	(104)

Triples is implemented by using array data structure from Hence having fixed size, to avoid this draw back linked list format of "Triples" notation is used now known as "Indirect Triples"

Scanned by CamScanner

Following are different type of translator present in language processing system

Preprocessor

→ Preprocessor is a translator that translates high level language into preprocessor high level language.

→ In C language all # statements are executed by the preprocessor.

Assembler

→ Assembler is a translator that translates assembly language into, relocatable, m/c code.

→ Relocatable m/c code is not executable code bcz linking is not done.

→ Linker is a program that makes different relocatable modules into single model module by resolving external references

Loader

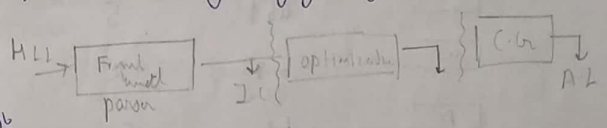
Loader is a program that loads executable code from secondary memory to memory to move them in memory.

Interpreter

→ Interpreter is a language processor that performs line by line execution of high level language.

→ One complete scan of source program is known as one pass of the compiler

→ In pass 1, HLL is translated into Intermediate code. In pass 2 code optimization is performed. In pass 3, assembly language is generated.



Gate 1996

Functions performed in pass 1 and pass 2 of in 2 pass assembler

Pass 1

i) Assign addresses to all stmnts in the prg
ii) Save the values assigned to all labels for use in pass 2
iii) perform some processing of assembler directives

Pass 2

i) Assemble instructions
ii) Generate data values defined by Byte, Word etc.
iii) Perform processing of assembler directives not done during pass 1
iv) Write the program and the assembling listing

Scanned by CamScanner

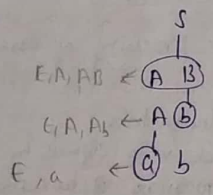
Viable Prefix

Set of prefixes in the right sentential form that appears immediately L.H.S part of handle known as viable prefix.

Viable prefixes are nothing but stack content in LR parser.

Stack	Input	Action
$	ab$	S
a	b$	R
A	b$	S
Ab	$	R
AB	$	R
S	$	accept

$S \to AB$
$A \to a$
$B \to b$



E, AB ← (A B)
E, AB ← A (b)
E, a ← (a) b

Whenever we get handle apply prefix oph.

Runtime Environment

Static Storage Allocation

→ In static storage allocation compiler takes decision regarding run time memory by looking at program text only

→ In static storage allocation recursive programs can not be executed and dynamic data structures can not be created.

→ Ex:- FOTRAN language

Dynamic Storage Allocation

→ In dynamic data storage allocation, compiler divides run time memory into code area, data area, stack area and heap area.

→ The generated m/c code is loaded into code area

→ Space for global variables allocated in a data area

→ Corresponding to every fun call one activation record is created and it is pushed into stack space

Scanned by CamScanner

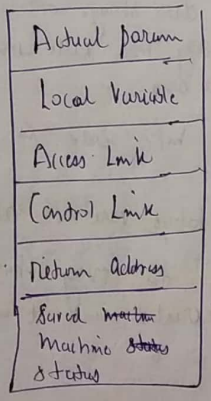
→ Whenever fun^n returns its value then activation record is popped out from stack. Hence allocation and deallocation of memory is completely automatic in stack space.

→ If we require any value throughout the program, that value can not be kept in stack space. Hence space is allocated in Heap area.

→ If the space is allocated in heap to deallocate memory explicitly in programs like garbage collection (or) free fun^n is used.

→ By using dynamic data storage allocation recursive programs can be executed and dynamic data structures can be created.

Activation Record



→ Access links points to global variables

→ Control link creates a link blw calling fun^n and called fun^n

→ by using access link static scoping is performed

→ by using control link dynamic scoping is performed.

Test (Revision)

* if we apply L-attributed evaluation for the S-attributed grammar then o/p will be same as S-attributed evaluation

* min no of nodes and edges in DAG representation

$a = b + c$
$c = a + d$
$d = b + c$ redundant
$e = d - b$
$a = e + b$

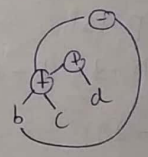
$\Rightarrow$

$a = b + c$
$c = a + d$
$e = a - b$
$a = a - b + b = a$ redundant

$\Rightarrow$

$a = b + c$
$c = a + d$
$e = a - b$

nodes = 6
edges = 6



→ Every LL(1) grammar must be
i) Unambiguous
ii) not left recursive
iii) left factored

Scanned by CamScanner

Revision

CFL & PDA

Q 2.34 P-150

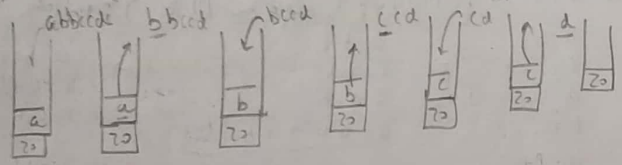
Sol. L(G) = lang recognized by ambiguous grammar
L(D) = lang recognized by ~~un~~ disambiguous grammar
L(G) = L(D)

Q 2.65 P-154

Sol. $\{a^m b^n c^p d^q / m+p = n+q, m,n,p,q \geq 0\}$

It is a CFL

Ex: abbccd



Q 2.39 P-157

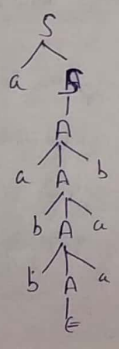
Sol: A-3, B-1, C-4, D-2 (Revise) Imp

Q 2.42 - 2.43

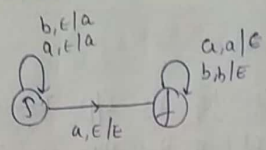
Sol. S → aS/A
A → aAb/bAa/ε

2.42) d) aabbaab

2.43) 6 steps 1 parse Tree



2.19 ~~2.49~~
Q ~~2.44~~
Sol.
P-148

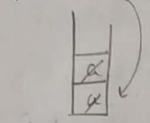


Here ε means read nothing from stack and it is not the stack bottom symbol.

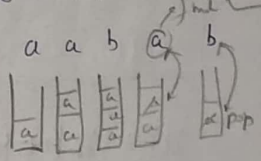
L = wab^n $w \in \{a,b\}^*$ $n < |w|$.

for all a's & b's push 'a's except for last 'a' which is used for transition to final state and pop.

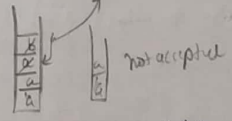
a) aaa — final



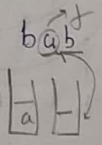
b) aabab



c) baaba — f



d) bab — f (f, a, ε)
invalid
not accepted



TM & Decidability And Undecidability

Q 3.39 P-168

Sol: $L_1 = \{\langle M \rangle \mid M$ takes at least 2016 steps on some i/p$\}$
$L_2 = \{\langle M \rangle \mid M$ take at least 2016 steps on all i/ps$\}$

For L_1 we can construct TM in which no. of steps are restricted to 2016. Hence TM will run only for finite # of steps on all i/p strings.

Scanned by CamScanner

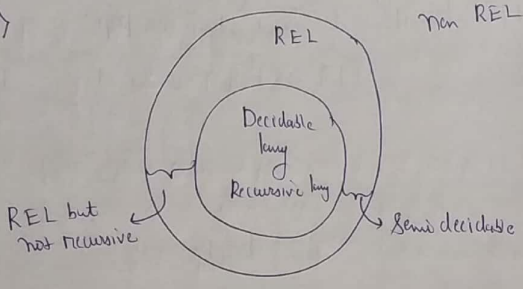
Therefore L_2 is recursive

If we can construct TM for all i/p ~~which~~ strings which takes atleast 2016 steps, we can use same TM for some i/p strings also.

Hence L_1 is also recursive [Create Overflow]

Test

i)



Undecidable

REL (or) non REL

ii) If TM i/p tape length is restricted to i/p length ~~of the language~~

Then The language accepted by TM is CSL

I) $(l^* m^* n^*)^* = (l m^* + m n^* + n l^*)$

II) $(mn + m)^* m = m(nm + m)^*$

III) $(l^* m^* n^*)^* \neq (l^* + m^* n + n^*)^*$

IV) $(l^* m)^* \neq (l + m)^*$

↖ here we can not get m separately without n

→ DPDA with empty stack will accept only those DCFL's which follow prefix property

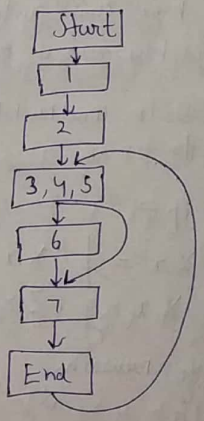
→ DPDA with final state method will accept ~~only those~~ All all DCFLs.

Control flow Graphs

```
1 i=10
2 j=1
3 a=i*j
4 b=i+j
5 if b ≤ a goto 7
6 a = a+1
7 i=i-1 goto 3
```

Nodes = 7

edges = 8



Scanned by CamScanner

$X = A/b$
$Y = C+D$
$Y = Y-X$
$X = D+Y$
$Z = F+Y$
$Z = Z+A$

Static Single Assignment

~~$t_1 = A$~~
~~$t_2 = b$~~
$t_3 = t_1 / t_2$
$t_4 = t_5 + t_6$
$t_7 = t_4 - t_3$
$t_8 = t_6 + t_7$
$t_9 = t_{10} + t_7$
$t_{11} = t_9 + t_1$

→ No of Unfilled Entries

$LR(0) \leq SLR(1) \leq LR(1)$

→ SDT with only synthesized attributes ~~does~~ does not have any cyclic dependency. So always have order of evaluation

Symbol table is constructed during analysis part of compiler

→ Consider SDT

$X \to XY \{ X.x = f(X.x, Y.y) \}$
$X \to Z \{ X.x = g(Z.z) \}$

Remove Left recursion

$A \to A\alpha / \beta$
$A \to \beta A'$
$A' \to \alpha A' / \epsilon$

$\alpha = Y \{ X.x = f(X.x, Y.y) \}$
So, $X \to Z \{ X.x = g(Z.z) \} X'$
$X' \to Y \{ X.x = f(X.x, Y.y) \} X' / \epsilon$
~~$X' \to \epsilon$~~

→ There exists a non-deterministic CFL whose reversal is DCFL — True

$L = \{ a^n b^n c \} \cup \{ d a^n b^{2n} d \}$

$L^R = \{ c b^n a^n \} \cup \{ d a^n b^{2n} d \}$

There exists a non-regular CFL whose Kleen closure is regular — True

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

$L^* = (a^p)^* \to$ regular

→ $X = \{ w / w \in \{0,1\}^*, n_0(w) = n_1(w) \}$
$Y = \{ w / w \in \{0,1\}^{49}, n_0(w) = n_1(w) \}$

X = Set of binary (bit) strings of length 8 having equal no of 0's & 1's.

if n = even # of n length bit strings $= {}^{n}C_{n/2}$
n = odd # of n length bit strings $= 0$

$|X| = {}^{8}C_{4} = {}^{8}C_{4} = 70$ $\quad |X| =$ (cardinality of)

$|Y| = 0$

$|X| + |Y| = 70$ ← Ans

→ Constant folding is possible if expression has Constants in value of assignment. It can be applied on 3-address code

Scanned by CamScanner

→ $L = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \phi \}$

↳ Emptiness problem for TM. is
It is non-REL

→ Leaf node of a tree always have only inherited attribute. → False

Leaf node of a tree have synthesized attributes as the values are taken from symbol table.

- For a given CFG, SLR(1) and LALR(1) parsing table both will have same shift entries. ⇒ True

→ Over the unary alphabet, every ~~CFG~~ CFL is regular → True

Over unary alphabet, every CSL is regular → False

Ex:- $\{a^p / p \text{ is prime}\}$ → CSL but not regular

$\Sigma = \{a\}$

→ Reversal of DCFL is always CF → True

bcz every DCFL is CF, & CFL are closed under reversal. ✓

(2019)

MTI

$L = \left\{ a^{\left\lfloor \frac{m}{n} \right\rfloor} / m, n \geq 1;\ n < m \right\}$ both are regular

$L = \left\{ a^{m^n} / m, n \geq 1;\ n < m \right\}$

Scanned by CamScanner