16/02/24
FOC

If $L(\alpha) \subseteq L(\beta)$ [denoted as $\alpha \leq \beta$]
then $\alpha + \beta \equiv \beta$

If $L(\alpha^*) \subseteq L(\beta^*)$ then $\alpha^* \beta^* = \beta^*$
and $\beta^* \alpha^* \equiv \beta^*$

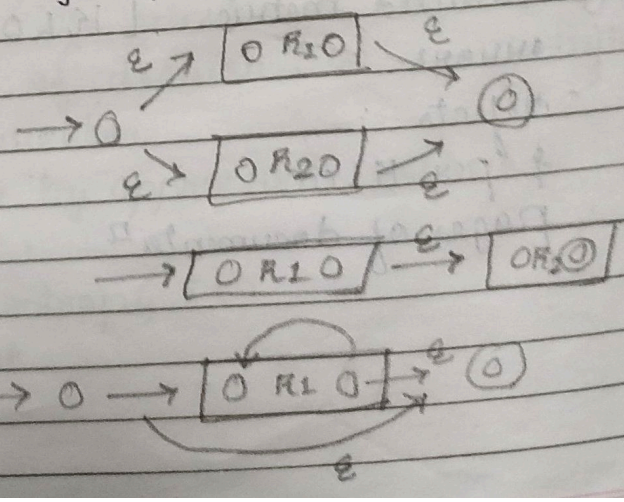
$a^*(a+b)^* \equiv (a+b)^*$

If $L(\beta) \subseteq L(\alpha^*)$ then $(\alpha+\beta)^* \equiv \alpha^*$
$\varepsilon + aa^* \equiv a^*$

$(\alpha\beta)^*\alpha \equiv \alpha(\beta\alpha)^*$
$(\alpha^*\beta)^*\alpha^* \equiv (\alpha+\beta)^*$
$\alpha^*(\beta\alpha^*)^* \equiv (\alpha+\beta)^*$

For any reg exp. R, there is an FA (M) st. L(M) = L(R)

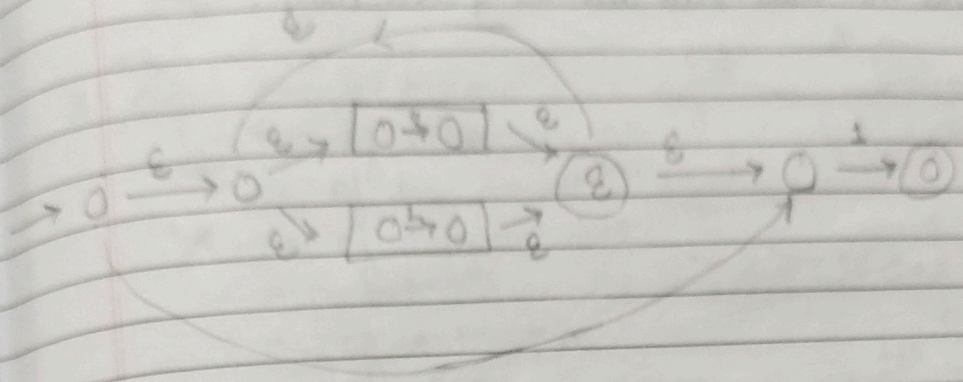
Basis: ϕ, ε, and a are reg exp. for any $a \in \Sigma$
Induction:
If R_1, R_2 are reg exp, they are:
$R_1 + R_2$
$R_1 \cdot R_2$
R_1^*
(R_1)



victue others
- ... on developing virtuous

stan

$(0+1)^* 1$ [Inductive approach]



given FA to reg exp.

↑

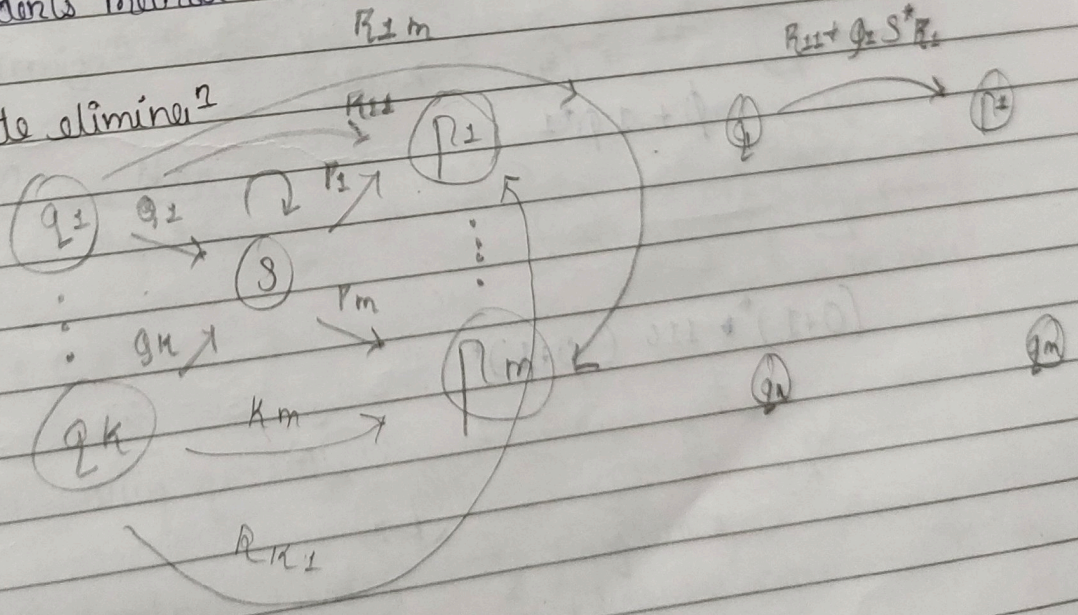
make generic, such that labels of R are not general symbols.
make label of arcs to strings

~~state~~ elimina^n method
ardents method

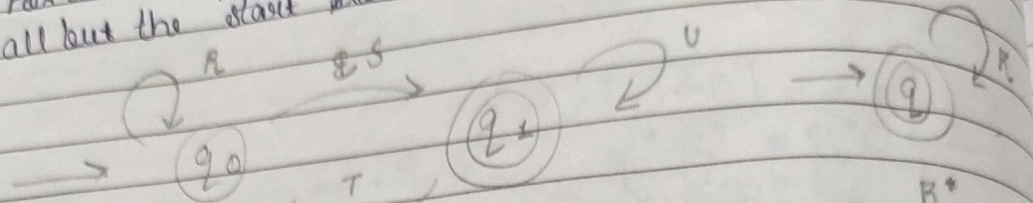
$= L(R)$

$\in \Sigma$

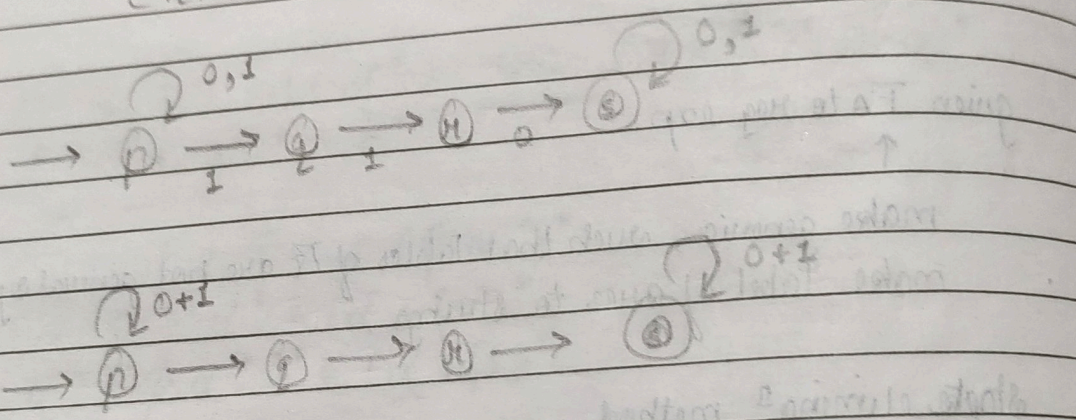
~~state~~ elimina^n



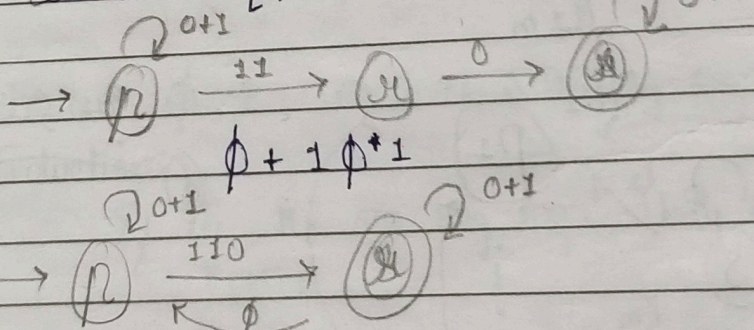
For each accepting state q, use this method to eliminate all but the start state and the q state itself.



$(R+SU^*T)^*SU^*$

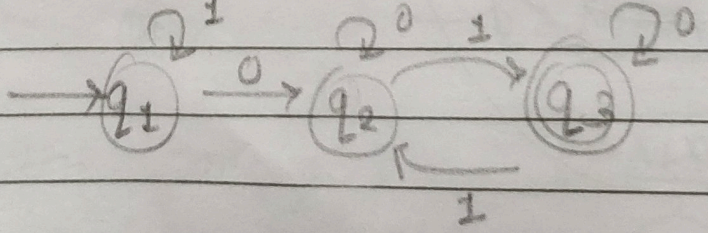


eliminate q, r



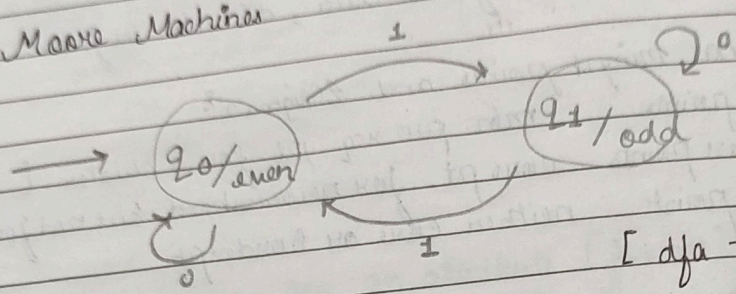
$\phi + 1\phi^*1$

$(0+1)^* \cdot 110\ (0+1)^*$



23/01/24
TOC

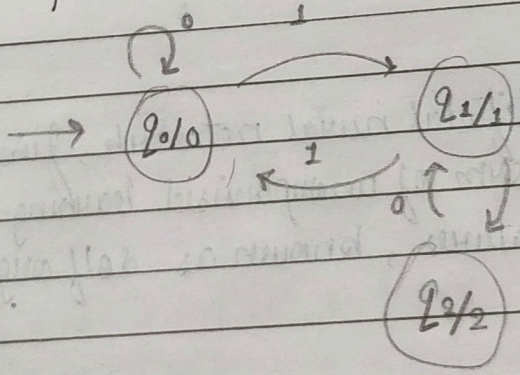
Moore Machines



[dfa to determine wether no. of 1's is even].

Q
Σ
Γ set of all symbols $\{E, O\}$
δ
λ $Q \to \Gamma$
q_0

[Output remainder when number is divided by 3]



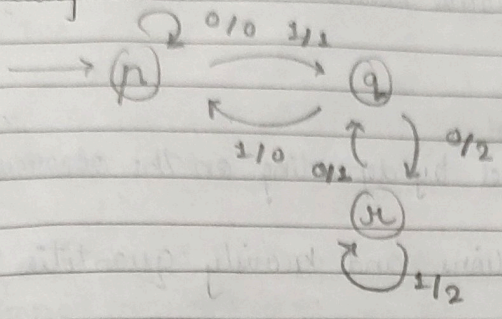
$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$\& \Gamma = \{1, 2, 3\}$
$\lambda : Q \to \Gamma$

CS	ns 0	ns 1	op
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

virtue others.
... on developing virtuous

69

Mealy Machine



Length of output string is K+1 for the case of moore machine

1 0 0 1 1 0
p q r q p q r
0 1 2 1 0 1 2

Counts no. of ~~blank~~ string 'aba' comes
total states if converting to moore mc.
no. of state · no. of o/p symbols

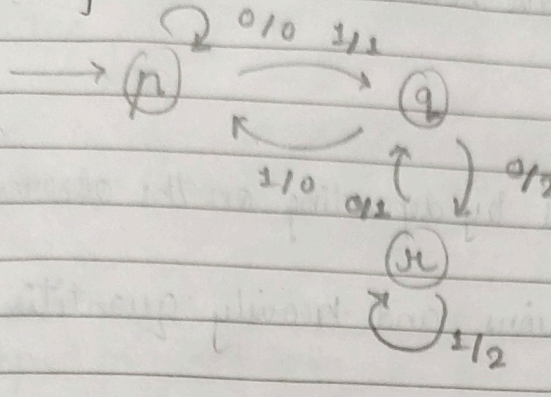
$L = \{ 0^n 1^n \mid n \geq 0 \}$ is not reg }

suppose L is reg, $M = \{ Q, \Sigma, \delta, q_0, F \}$ &
a dfa such that $L(M) = L$

let $|Q| = K$

pumping lemma
Suppose L is regular, find $M = (Q, \Sigma, \delta, q_0, F)$ is
a dfa, st. $L(M) = L$
let $|Q| = K$ consider any $x \in L$, $|x| \geq K$
$x = a_1 a_2 a_3 \ldots a_m$ $m \geq K$

Mealy Machine



Length of output string is K+1 for the case of moore machine

1 0 0 1 1 0
p q r q p q r
0 1 2 1 0 1 2

Counts no. of ~~blank~~ ~~times~~ string 'aba' comes
total states if converting to moore mc.
no. of state · no. of o/p symbols

$L = \{ 0^n 1^n \mid n \geq 0 \}$ is not reg }

Suppose L is reg, $M = \{ Q, \Sigma, \delta, q_0, F \}$ &
a dfa such that $L(M) = L_1$

let $|Q| = K$

pumping lemma
Suppose L is regular, find $M = (Q, \Sigma, \delta, q_0, F)$ is
a dfa, st. $L(M) = L$
let $|Q| = K$ consider any str $x \in L$, $|x| \geq K$
$x = a_1 a_2 a_3 \dots a_m$ $m \geq K$

on developing virtuans

11

26/02/24
ML TOC

L is reg, it satisfies property P(L)
¬P(L) → L is not reg.

If L is regular, then any string that is long enough can be pumped in, resulting in new longer string, all of them being in the language.

all finite languages are regular.
- no. of strings are finite
- planked and lead to accepting state

pumped substring will of length non zero.

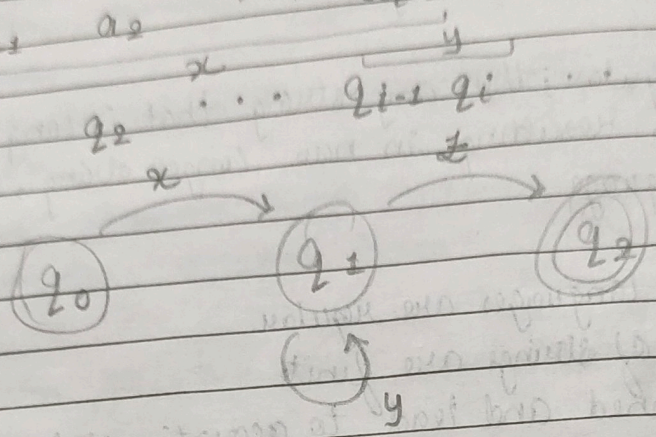
pumping lemma for reg lang. ∃ p
let L be a reg. lang. Then, there is a constant, depending on L, such that for any string s in L, with |s| ≥ p, it can be broken into three pieces as s = xyz, satisfy following condⁿ:

(1) |xy| ≤ p ∧
(2) |y| > 0 ∧
(3) for each i ≥ 0, $xy^iz \in L$. ∧

$$\exists p \left[\forall s \left[\exists xyz \;\; 1, 2, 3 \right] \right]$$

Theory of working time

$M = (Q, \Sigma, \delta, q, F)$
let $p = |Q|$
consider a string $s = a_1, a_2 \dots a_n$, $n \geq p$

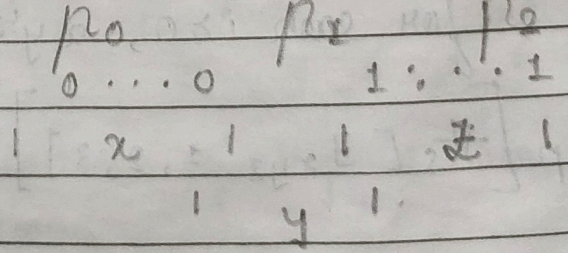


Lemma satisfaction doesn't prove language regularity. pumping lema doesn't help you same.

prove that $L = \{0^n 1^n \mid n \geq 0\}$ is not reg. using PL.
proof by contradiction

suppose L is regular, then it must satisfy PL.
let p be the const. pl.

consider $s = 0^p 1^p$, $p \geq 0$, clearly, $s \in L$ and $|s| = 2p \geq p$



Case - I : let $y = 0^k$, $k \leq p$, $k > 0$

$s = 0^p 1^p = xyz = 0^{p-k} 0^k 1^p$

$x \quad y \quad z$

let $i = 2$ & $\notin L$.

Case II : let $y = 1^k$

Case III : let $y = 0^k 1^l$

prove that $L = \{ x \in \{0,1\}^* \mid x$ has equal no of 0's & 1's$\}$ is not reg lang!

proof by contradict

pumping lemma

consider $s = 0^p 1^p$, $s \in L$, $|s| = 2p$.

$\underbrace{0 \cdots 0}_{p} \quad \underbrace{1 \cdots 1}_{p}$

$\vdash xy \dashv \vdash z \dashv$

rule 1: $|xy| \leq p$

as $|xy| \leq p$, y will only contain 0's.

let $y = 0^k$, $0 < k \leq p$

$s = 0^{p-k} 0^k 1^p$

$xy^2z = 0^{p+k} 1^p \notin L$

$L = \{ x \in \{0,1\}^* \mid x = ww, w \in \{0,1\}^* \}$

$0^p 1 \; 0^p 1 \leftarrow$ $0^p 0^p x$

$\uparrow$

$00 \; x$

27/02/24
ML

K means unsupervised
K nn supervised

Lazy Neighbour
K-nearest neighbour
- computationally expensive
- should be normalized else high range variables can bias it
- works on pre processing stage, like noise removal

advantages & disadvantages
for numerical data - use euclidean distance
for binary data (yes/no) - use hamming distance @
same - 0, different - 1
same values - first come first serve.
finding K : either given or sqrt (total values) or n-1
equal length example @

29/02/24
TOC

L = { 0^n | n is a prime number } is not a regular languages.

Context Free Languages [CFL]
- has recurrsive notation known as Context Free Grammar
- a generating device, used in compilers parser.
[automata are recognizing device]
↑ structure translating device

also used in document type descrip^z in xml.

L palin = { w ∈ {0
- binary palind
recursive defina

Basis: ε, 0 an
Induction: If

Grammar: {
P → ε
P → 0
P → 1
P → 0P0
P → 1P1
}

Context
- a

k =

Virtue ethics
- focus on developing virtuous

75

stay

Lpalin = { w ∈ {0,1}* | w = w^R } prove it is not reg.
- binary palindromes. using PL.
recursive defin^n [basis induction]

Basis: ε, 0 and 1 are in Lpalin
Induction: If w ∈ Lpalin then
- 0w0 ∈ Lpalin
- 1w1 ∈ Lpalin

Grammar: {
P → ε
P → 0
P → 1
P → 0P0
P → 1P1
}

set of variables
set of terminating symbols [terminals]
start symbol
set of productions
set of rules = Productions
head → body

grammar

Context Free Diagram:
- a cfg, g is 4tuple (V, T, P, S) where,
V: set of variables or non terminals P
T: set of terminals {0, 1}
P: set of 'product^n of the form.
A → α where A ∈ V and
α ∈ (V ∪ T)*
S: S ∈ V, start symbol. S = P

L = {

RL ⊂ CFL

capitals → vars.
small → terminals

Basis: a is an identifier, b is an identifier
Induction: If I is an identifier, then
- Ia
- Ib
- I0
- I1

are also identifiers, is reg. language

$I \to a \mid b \mid I0 \mid I1 \mid Ia \mid Ib$

Any regular ~~exp~~ language can be given context free grammar and is context free language. Vice versa is not true.
example: Palindrome

Basis: I is an expression
Induct: If E_1 and E_2 are expression, then so are:
$E_1 + E_2$
$E_1 * E_2$
(E_1)

$E \to I$
$E \to E+E$
$E \to E*E$
$E \to (E)$

$V = \{E, I\}$
$T = \{a, b, 0, 1, +, *, (,)\}$
P —
$S = \{E\}$

each var. have their own lang.
language of a variable is language of over grammar [terminal]

$I \to a \mid b \mid I0 \mid I1 \mid Ia \mid Ib$
$E \to I \mid E+E \mid E*E \mid (E)$

Set of all terminal str generated grammar

01/03/2024

Basis: For any $\alpha \in (V \cup T)^*$ $\alpha \Rightarrow^* \alpha$

Induction:

If $\alpha \Rightarrow^* \beta$ and $\beta \Rightarrow^* \gamma$ then $\alpha \Rightarrow^* \alpha\gamma$

$\Rightarrow^*$ - one or more derivation

a^* $\quad A \to \varepsilon \qquad A \to \gamma$

$A \to aA \qquad \alpha A \beta \Rightarrow \alpha \gamma \beta$.

$A \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow aaa$

$A \Rightarrow^* aaa$

Let $G = (V, T, P, S)$ be a cfg, then the language associated with G, denoted as $L(G)$

$$L(G) = \{ x \in T^* \mid S \Rightarrow^* x \}$$

10101

$P \Rightarrow^* 1P1 \Rightarrow^* 10P01 \Rightarrow^* 10101$

$L(G_{palin}) = L_{palin}$

Consider any $k+1$ length str.

Case-1 Let $|w| = k+1$, $k \neq 1$

Let w, $0x0$ Lets generate w as

$P \Rightarrow 0P0 \Rightarrow^* 0x0$.

design a grammar for $a^n b^n$

ε, ab, aabb, aaabbb.

$A \Rightarrow^* aAb$

$\{a^i b^j, i = 2j\}$

ε, aaab, aaaaabb

$A \Rightarrow^* \varepsilon$

$A \Rightarrow^* aaAb$

$\{a^i b^j c^k, j = i+k\}$

ε, abbc, aabbbbcc

$a^i b^j c^k = a^i b^i \{b^k c^k$

$A \Rightarrow^* \varepsilon$ $\quad A \Rightarrow^* aAb$

$AB \Rightarrow^* aAbbBC$ $\quad B \Rightarrow^* bBC$.

$S \Rightarrow^* AB$

Generating
ab bbcc

$A \Rightarrow^* aAb\ bBC$
$A \Rightarrow^* ab\ bbBCC$
$A \Rightarrow^* ab\ bbcc$

} go step by step.

context free languages are closed under concatenation

a language is context free if there is a cfg generating to it.

If L_1 and L_2 are cfl then so, L_1 & L_2 ..

let $G_1 = (V_1, T, P_1, S_1)$ & $G_2 = (V_2, T, P_2, S_2)$ be two cfgs such that $L(G_1) = L_1$ and $L(G_2) = L_2$ we can construct $CFG\ G_c = (V_c, T, P_c, S_c)$ as follows such that $L(G_c) = L_1 L_2$.

20+60+20+60+20
+20

40,000
45,000p / 10 = 4500 = 200 · 4 = 800
or
250 = 1000

$V_e = V_1 \cup V_2 \cup \{s_1\}$, $se \; V_1 \cup V_2$
$S_e = S$
$P_e = P_1 \cup P_2 \cup \{S \rightarrow s_1 \cup s_2\}$

If L_1 and L_2 are cfl, so is $L_1 \cup L_2$.

$G_1 = \{V_1, T, P_1, S_1\}$
$G_2 = \{V_2, T, P_2, S_2\}$
$G_u = \{V_u, T, P_u, S_u\}$

$V_1 \cap V_2 = \phi$
$S_1 \cup S_2 = S$

04/03/24
ML.

Markov Property:
- used in Markov chain
- generate probablity from probablity matrix [graph]

Six components of hidden markov model [hmm]

problems in hidden markov model:
- evaluation problem → solved by forward algorithm
- decoding problem → solved by viterbi algorithm
- learning problem → solved by Baum welch algorithm

// hmm example for weather forcasting
if x today, x / y / z tommorow, next to next for day after tommorow or next two days.

04/08/24
ToC

$a^i b^j c^k$, $j = i+k$
$= a^i b^i b^k c^k$

$S \to AB$
$A \to aAb \mid \varepsilon$
$B \to bBc \mid \varepsilon$

derivation of abbbcc

$S \Rightarrow_{lm} AB \Rightarrow_{lm} aAbB \Rightarrow_{lm} abB \Rightarrow_{lm} abbBc$
$\Rightarrow_{lm} abbbBcc \Rightarrow abbbcc$

"there exists a rm derivation of a language iff there exists a lm derivation."

parse tree
generation of string in a given graph.
- internal node - variables
- terminal symbols & start symbol as root node.

yield of parse tree:
concatenation of leaf nodes from left to right.

root may or may not be a start symbol.
to be part of a language, root node should be start symbol, leaf node should be terminal, or epsilon or variables also.

" if there left most derivation, right most derivation for a given string, there exists parse tree for it."
- vice versa is also true.

04/03/24
ToC

$a^i b^j c^k$, $j = i+k$
$= a^i b^i b^k c^k$

$S \to AB$
$A \to aAb \mid \varepsilon$
$B \to bBc \mid \varepsilon$

derivation of abbbcc

$S \Rightarrow AB \overset{lm}{\Rightarrow} aAbB \overset{lm}{\Rightarrow} abB \overset{lm}{\Rightarrow} abbBc$
$\overset{lm}{\Rightarrow} abbbBcc \Rightarrow abbbcc$

"there exists a rm derivation of a language iff there exists a lm derivation."

parse tree
generation of string in a given graph.
- internal node - variables
- terminal symbols & start symbol as root node.

yield of parse tree:
concatenation of leaf nodes from left to right.

root may or may not be a start symbol.
to be part of a language, root node should be start symbol
leaf node should be terminal, or epsilon or variables also

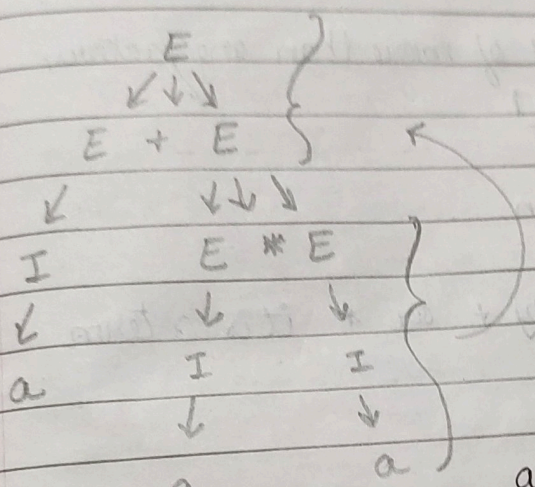
" if there left most derivation, right most derivation for a given string, there exists parse tree for it"
- vice versa is also true.

virtue ethics:
- ... on developing virtuous

$I \rightarrow a|b|Ia|Ib|I0|I1$
$E \rightarrow I|E+E|E*E|(E)$

$a+a*a$

$V = \{E, I\}$
$T = \{0, 1, a, b, +, *, (,)\}$



precidence & associativity

ambigious grammer.
- more than parse tree for

a grammer is ambigious if there is a string
$G = (V, \hat{I}, P, S)$ x in the grammer
$x \in L(G)$

every string may not have two parse tree. that has two distinct parse trees.

" grammer is ambigious if there is a string x for which there are two left most derivations."
or right most.

$a+a*a$

$E \rightarrow E+E \Rightarrow I+E \Rightarrow a+E \Rightarrow aE*E \Rightarrow a+I*E$
$\rightarrow a+a*E \Rightarrow a+a*I \Rightarrow a+a*a$

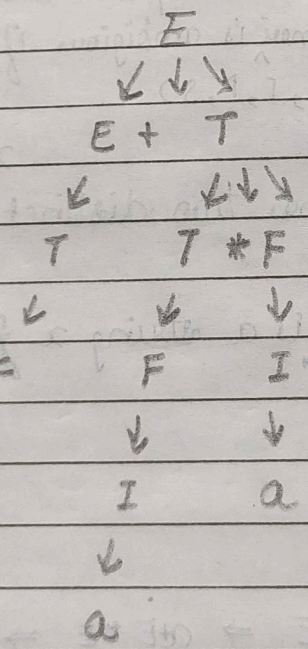
$E \rightarrow E*E \Rightarrow E+E*E \Rightarrow aI+E*E \Rightarrow a+E*E \Rightarrow a+I*$
$\Rightarrow a+a*E \Rightarrow a+a*I \Rightarrow a+a*a.$

$F \to I$
$T \to F \mid T * F$
$E \to T \mid E + T$

factors cannot be broken by any adjecent operation +, *
$I \to (E)$ ex. : $(a * a) + b$

term can be broken by * but not +
↑
it can be a factor, or product of more than one factors.
ex. $a + \underbrace{[b * c]}$
↑
broken.

Expression can be broken by + or * it is a term or sum of terms.



"there are some languages where the grammer would be ambigious"
↑
always
"unambigious grammer is not possible."

- Inherently ambigious language.

" If a language is ambigious, the grammer is bound to be ambigious, but the vice versa may or may not be true."

05/03/24
ML

Support vector machine
- length of margin [d+ & d-]
- should be max. to avoid misclassification.

tuning parameters
- regularization [↓ - less error, ↑ more misclassification]
- gamma [↓ - far pt to be considered, ↑ near pt.]
- kernal [linear, polynomial, exponential] (trick)
- margin [soft margin, hard margin]

07/03/24
TOC

$L_1 = \{a^n, b^n, c^m, d^m, n, m \geq 1\}$
$L_2 = \{a^n, b^m, c^m, d^n, n, m, \geq 1\}$

$A \rightarrow aAb \mid ab$ $\quad S_1 \rightarrow AB$
$B \rightarrow bBd \mid cd$

~~$C \rightarrow bCc \mid bc$~~ $\quad C \rightarrow aCd \mid aDd$
~~$D \rightarrow aCd \mid ad$~~ $\quad D \rightarrow bDc \mid bc$

$S_u \rightarrow S_1 \mid C$

$S \Rightarrow S_1 \Rightarrow AB \Rightarrow abB \Rightarrow abcd$
$S \Rightarrow C \Rightarrow aDd \Rightarrow abcd.$

2 grammers giving same string, are ambigious

balanced parenthesis & if else statements

push down automata
= nfa-ϵ, with a stack.
- arbitarily large number of symbols.
- push/pop on stack.

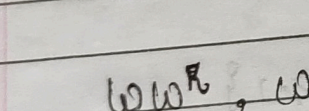
$\delta(q, a, X) = (p, \alpha)$ where $a \in \Sigma \cup \{\epsilon\}$, $\alpha \in \Gamma^*$, Γ = stack alphabet

q = initial state, X = top of stk, p = next state, α = string

$ww^R, w \in \{0,1\}^*$

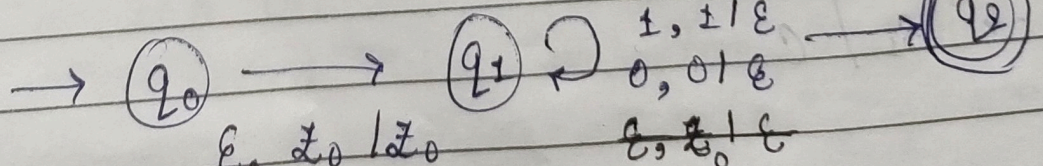
7 tuple

Q: set of finite state
Σ: set of finite symbols
Γ: set of stack symbols
δ: $Q \times \Sigma \cup \{\epsilon\} \times \Gamma$
q_0: start ~~start~~ state
Z_0: start stack symbol
F: final state; $F \subseteq Q$.



current top next
1, 1 | 11
1, 0 | 10
1, Z_0 | 1Z_0

0, 1 | 01
0, 0 | 00
0, Z_0 | 0Z_0



ϵ, Z_0 | Z_0
ϵ, 1 | 1
ϵ, 0 | 0.

$L(P) = \{ x \in \Sigma^* \mid (q_0, x, z_0) \vdash^* (p, \varepsilon, \alpha) \}$

for any $p \in T$ and $\alpha \in \Gamma^*$

basis: for any id I, $I \vdash^* I$ (zero step)

induc^n: if $I_1 \vdash^* I_2$ and $I_2 \vdash^* I_3$, $I_1 \vdash^* I_3$

sequence of ids for a pda

1111

$(q_0, 1111, z_0)$

$(q_0, 111, 1z_0)$ $(q_1, 1111, z_0)$

$(q_0, 11, 11z_0)$ $(q_2, 1111, z_0)$ ×

$(q_0, 1, 111z_0)$ $(q_1, 111, 1z_0)$

$(q_1, 11, 11z_0)$ $(q_1, 11, z_0)$

$(q_1, 1, 111z_0)$ $(q_2, 11, z_0)$ ×

$(q_1, \varepsilon, 11z_0)$

final end can be of the stack anyth.

→ accepting a string by entering into final state

→ accepting a string by emptying the stack completely

virtue ethics

... developing virtuous

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

- $L(P) = \{ w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \alpha) \}$, $p \in F, \alpha \in \Gamma^*$

- $N(P) = \{ w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon) \}$, $p \in Q$.

$C_F \cup C_N = \{$ set of context free lang. $\}$

$C_F \subseteq C_N$

$C_N \subseteq C_F$ if for some lang. L_1, $\{ L_1 = L(P)$ for some pda$\}$ then there is a pda P_n such that $L_1 = N(P_n)$

12/03/24

TOC

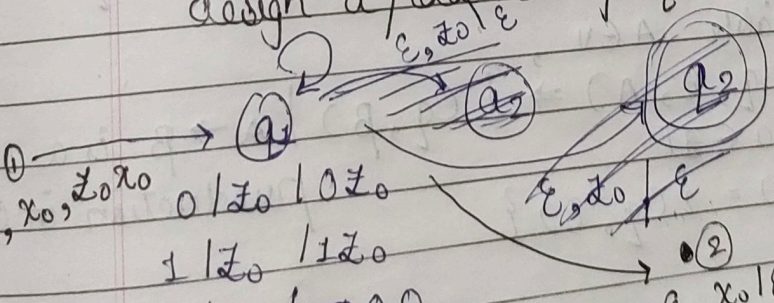
Let $P_n = (Q_n, \Sigma, \Gamma, \delta_n, q_n, Z_n)$

Let $L_1 = N(P)$

We will construct $P_f = (Q_f, \Sigma, \Gamma, \delta_f, q_f, Z_n)$

0110

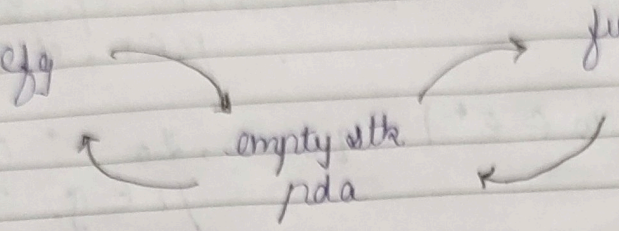
design a pda having equal no of zeroes and ones



Date

grammar - generating device
pda - recognizing device

A lang L_x is cfl iff there is a context free grammar G such that $L(G) = L_x$.



sentential form (for any grammar $G = (V, T, P, S)$)

- α is said to be a sential form if $S \Rightarrow^* \alpha$

left & right sentential form, depending upon form of deriving

Let $G = (V, T, P, S)$, construct pda $P_n = (Q_n, \Sigma, \Gamma, \delta, q_0, Z_0)$ such that $N(P_n) = L(G)$

Consider
$x A \alpha$ as a sentinal form where
$x \in T^*$, $A \in V$, $\alpha \in (V \cup T)^*$

here, we call $A\alpha$ as the tail of the 2.s.f

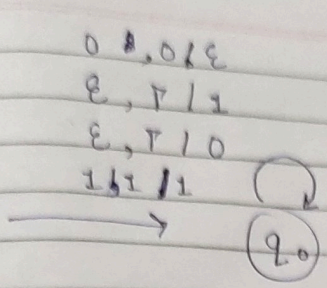
- for any variable $A \in V$,
$\delta(q, \varepsilon, A) \Rightarrow \{(q, \beta) \mid A \to \beta$ is a production in $P\}$

for every terminal $a \in T$
- $\delta(q, a, a) = \{(q, \varepsilon)\}$

virtue others.

... developing virtues

stay



(q, 01.0, p)

$Q = \{q\}$

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

$\Gamma = V_{OT} = \{\perp, 0, 1\}$

$q_0 = q$

$Z_0 = \perp$