DOMS | Page No.
Date 01/02/2022

L-20

Outline (Reglang ⊆ Σ^*)

- Lang. generators
- RegEx
- Grammar, context-free, examples
- Terminology $\Rightarrow_G^*$, $\Rightarrow_G$, derivation, L(G)
- Examples.

So far,

→ Language Acceptors: Finite Automata, checks if w ∈ L or not
(Notion of membership)

→ Regular Expressions: Notation to represent language using symbols. (Finite representation via symbols)
i.e.
Language Generators

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

Algo to generate

start → step① Generate o/p 'a'
step② o/p any no. of 'a's (#a's) (or) any no. of 'b's. (#b's)
step③ o/p 'b'
end ←

Let S be a word in Lang(r)., $r = a(a^* + b^*)b$
middle

definite pairs
$S \rightarrow a\ M\ b$
$M \rightarrow A$ (any #a's)
$M \rightarrow B$ (any #b's)
(or)

base / inductive
$A \rightarrow \epsilon$ zero a's
$A \rightarrow aA$ one 'a' + more (maybe) 'a's.

base / inductive
$B \rightarrow \epsilon$
$B \rightarrow bB$

DOMS | Page No.
Date / /

~~S → ab~~

~~S → aMb~~

Generate w=ab using the new notation.

$S \rightarrow aMb \rightarrow aAb \rightarrow ab$

(M → A) using rule 2

(A → ∈) using rule 4

using rule 1

Generate w=abbb.

$S \xrightarrow[\text{rule 1}]{} aMb \xrightarrow[\text{rule 3}]{} aBb \xrightarrow[\text{rule 7}]{} abBb \xrightarrow[\text{rule 7}]{} abbBb \xrightarrow[\text{rule 6}]{} abbb$

Try generating w=aabb

$S \xrightarrow[\text{rule 1}]{} aMb \xrightarrow[\text{rule 2}]{} aAb \xrightarrow[\text{rule 5}]{} aaAb \xrightarrow[\text{rule 4}]{} aab$ ✗

$aaAb \xrightarrow[\text{rule 5}]{} aaaAb.$ ✗

⇒ No paths generate aabb.

Jargon: "S → aMb" → Rules/Productions

Small letters → Alphabet (Σ) = Terminals → can not be replaced

Caps letters → Non-terminals (S, A, B) → (N) → can be replaced

Set of Rules = Grammar (?)

Vocabulary. V = N ∪ Σ

$G = (N, \Sigma, R, S)$

N → Non Terminals; Σ → Terminals; R → rules; S → start symbol

$\rightarrow \subseteq V^* \times V^*$ Relation on vocab.

- $Lang(G) = lang(r)$

- Applications of rules = "Derivation"

  $S \to\to\to w \in \Sigma^*$

  "w is derived from S"

- Derivation is a transitive + reflexive closure of the relation $\Rightarrow$, i.e., $\Rightarrow^*$

- language of a grammar, G,

  $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$

- ($\Rightarrow^*$ is $\to\to\to\ldots\to$) A sequence of individual rule-applications

Ex) $\left.\begin{matrix} S \to aSb \\ S \to \epsilon \end{matrix}\right\} = R$ $\quad G = (N, \Sigma, R, S)$.

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

ex ① ϵ: $S \xrightarrow{R_2} \epsilon$

ex ② ab. $S \xrightarrow{R_1} aSb \xrightarrow{R_2} ab \in L(G)$

ex ③ aabb : $S \xrightarrow{R_1} aSb \xrightarrow{R_1} aaSbb \xrightarrow{R_2} aabb \in L(G)$

- Grammars exist for regular, non-regular, etc.

- context-Free Languages:

  Languages generated by/derived from CFGs.

- CFGs:

  when in the derivation process, application of rules is never "dependent" on ~~productions~~ other letters "around"

$S \to aMb \to aAb$

$\to aBb$ don't care abt a, b when replacing M

DOMS | Page No.
Date / /

- Replacing is done w/o consideration of existing word environment (i.e, context)
- For each rule, (left: N / right: V^*) ($\Rightarrow : N \to V^*$) Rule format.

Ex) Language of well-balanced parentheses:

(((X))

$\Sigma = \{(,)\}$ $\epsilon, (), (()), ()()$

$R_1: S \to \epsilon$

$\begin{bmatrix} S \to S() \\ S \to ()S \end{bmatrix}$

$R_2: S \to (S)$

$R_3: S \to S \cdot S$

$()() :-$

$S \to S() \to (S)()$

$\downarrow$

$()()$

$(())(())$

S S ( )

Ex) Language of palindromes

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

Base: $S \to \epsilon$ R_1; $S \to 0$ R_2; $S \to 1$ R_3

Inductive: $S \to 0S0$ R_4; $S \to 1S1$ R_5

$N = \{S\}$

$R = \{R_1, R_2, R_3, R_4, R_5\}$

$S = S$.

$G = (N, \Sigma, R, S)$

- Rules, $R : N \to (NUT)^*$
- CFGs are word-generators.

DOMS | Page No.
Date 02/02/2022

L-21

Outline
- L_bal - CFG
- Example ()()
- Parse Trees ⇒ Defn, Yield
- Left/right-most derivations
- Ambiguous grammar
- Unambiguous grammar/lang
- [Regular ⊆ CFL's] (?)

Ex) $L_{bal} = \{ w \in \{(,)\}^* \mid w \text{ is well-balanced} \}$

$$\begin{bmatrix} R_1: & S \to \epsilon \\ R_2: & S \to (S) \\ R_3: & S \to S \cdot S \end{bmatrix} \quad \text{i.e. } S \to \epsilon \mid (S) \mid S \cdot S$$

$N = \{S\}$
$T = \Sigma = \{(,)\}$
$S = S$
$R = \{R_1, R_2, R_3\}$

Ex) $w = ()$

$$S \xrightarrow[R_2]{} (S) \xrightarrow[R_1]{} (\epsilon) = ()$$

Just order of application

One word may have multiple derivations

Ex) $w = ()()$

$$S \xrightarrow[R_3]{} S \cdot S \xrightarrow[R_2]{} (S) \cdot S \xrightarrow[R_2]{} (S)(S) \xrightarrow[R_1]{} (\epsilon)(S) \xrightarrow[R_1]{} ()() = ()()$$

Ex) $w = (())$

$$S \xrightarrow[R_2]{} (S) \xrightarrow[R_2]{} ((S)) \xrightarrow[R_1]{} ((\epsilon)) = (())$$

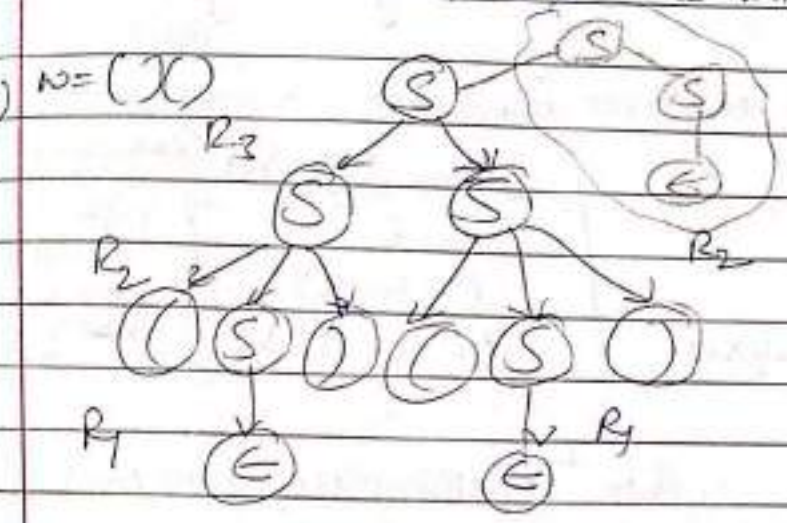
Ex) $w = (()$

$$S \xrightarrow[R_2]{} (S) \xrightarrow[R_2]{} ((S) \quad X$$

$$\xrightarrow[R_1]{} (() \quad X$$

→ collect all derivations that are interconvertible, call them equivalent.

Ex) $w = ()()$



[Parse Tree]
All equivalent derivations can be located in this.
leaves = terminals.

→ Leftmost derivation : Always replace leftmost nonterminal (like preorder)
⇓
Represents all equivalent derivations
$R_2 \to R_2 \to R_1 \to R_2 \to R_1$

→ Rightmost derivation : Always replace rightmost non-terminal (like inverse postorder)
⇓
Also represents all equivalent derivations
$R_2 \to R_2 \to R_1 \to R_2 \to R_1$
(R R L L R)

→ Other derivations : Extra ϵ-terms.

→ Yield of a tree = leaves from left to right, giving word represented by that tree.

→ Given a grammar G (for a bag), such that a word has multiple (≥2) parse trees, then the grammar is termed "AMBIGUOUS"

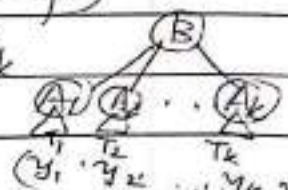
→ If all words have unique, single parse trees, then grammar is termed "NON-AMBIGUOUS."

→ Parse tree : Given a grammar G, (bottom-up)

① $a \in \Sigma$: (a)

② $A \to \epsilon$: (A → ϵ)

③ $B \to A_1 A_2 \cdots A_k$



→ Programming languages have unambiguous grammars.

→ For well-balanced parentheses language,

Ambiguous:

$R_1: S \to \epsilon$

$R_2: S \to (S)$

$R_3: S \to S.S$ (alphabet)

Unambiguous (case-by-case)

$R_1: S \to \epsilon$

$R_2: S \to (S)$

$R_3: S \to (S)S$

right hand side always has some terminals or ϵ.

claim: unambiguous grammar

→ There are languages having ambiguous / unambiguous grammars, but some don't have unambiguous grammars. ("Inherently Ambiguous Languages")

→ Is there a procedure/algorithm to tell if a language is inherently ambiguous?

No. undecidable. Outside of NP.

→ Are all regular languages in CFLs? Yes.

(Reglang $\subset$ CFL)

Pf. later.

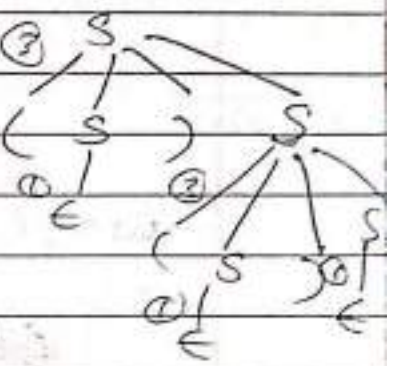
DOMS | Page No.
Date 03 / 02 / 2022

L-22

Outline

- Unambiguous Grammar
- Reglang $\subset$ CFL's
- PDA : $a^n b^n$, $w \Sigma w^R$, $w w^R$
- Reglang - PDA

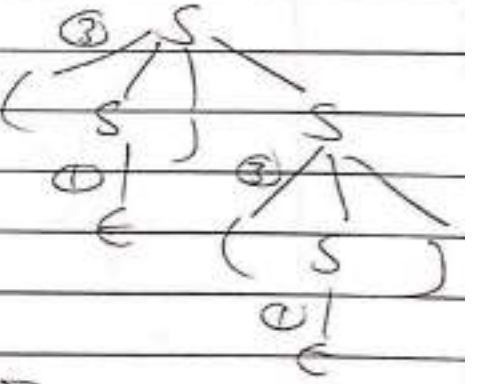
→ L_{bal} : $\begin{pmatrix} S \to \epsilon & ① \\ S \to (S) & ② \\ S \to (S) \cdot S \cdot & ③ \end{pmatrix}$ Is it ambiguous?



$w = ( )( )$

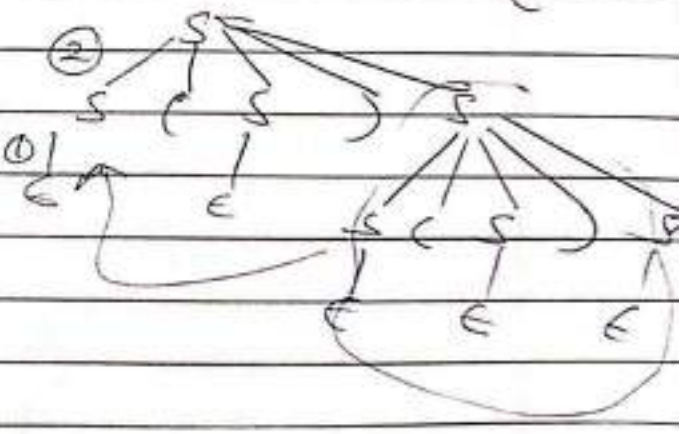
$S \xrightarrow{③} (S)S \xrightarrow{②} (S)(S) \xrightarrow{①} \xrightarrow{①} ( )( )$

$S \xrightarrow{③} \xrightarrow{③} \xrightarrow{①} \xrightarrow{①} \xrightarrow{④} ( )( )$



Hence, this word has 2 distinct parse trees ⇒ ambiguous

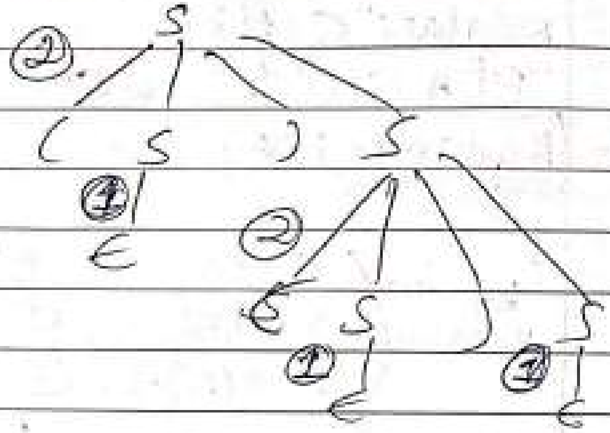
→ L_{bal} : $\begin{pmatrix} S \to \epsilon & ① \\ S \to S(S)S & ② \end{pmatrix}$



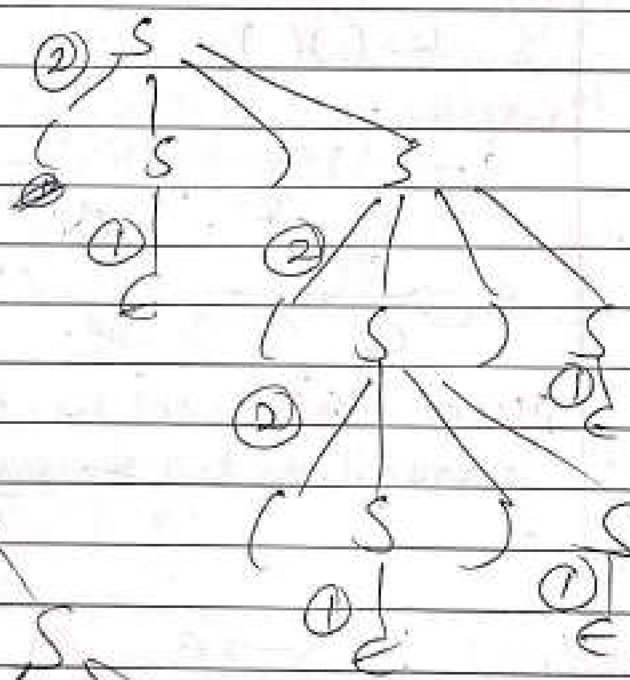
Still ambiguous!

→ L_{bal} : $\begin{pmatrix} S \to \epsilon & ① \\ S \to (S)S & ② \end{pmatrix}$

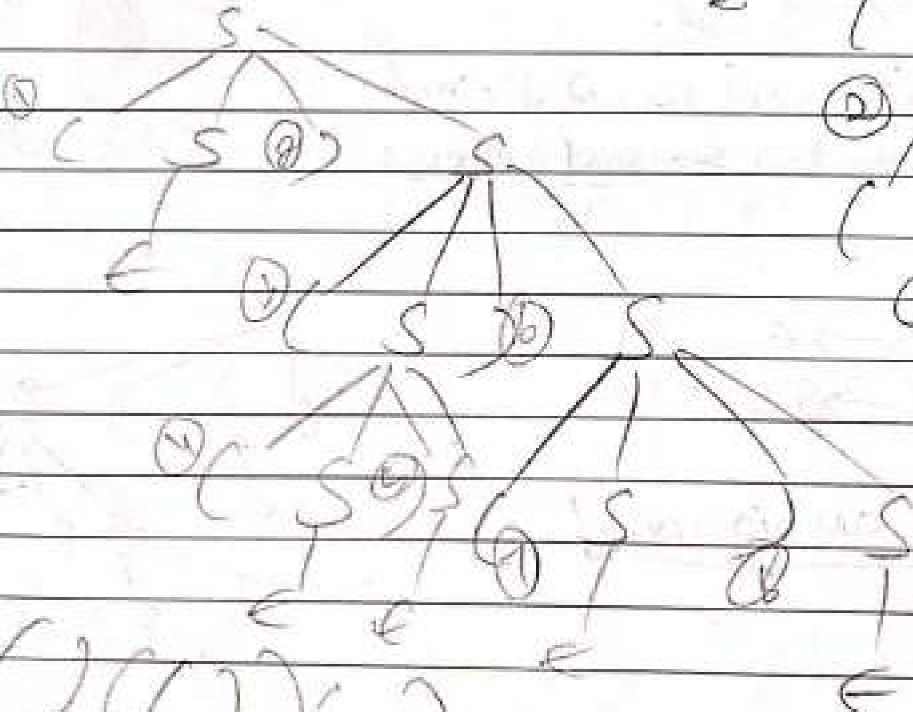
w = ()()



w = () (())



()(())()



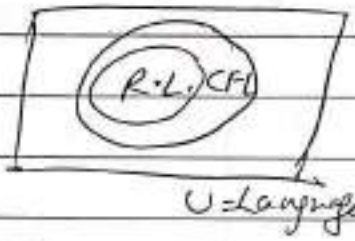
()(())()

Df. In Assignment Lol

DOMS | Page No.
Date / /

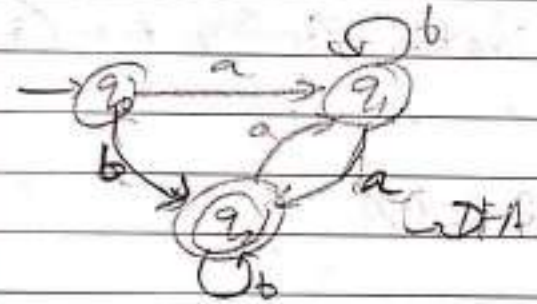
→ W.K.T. Reglang ∩ CFL ≠ ϕ
CFL − Reglang ≠ ϕ (Non Reg)
Reglang − CFL ≟ ϕ

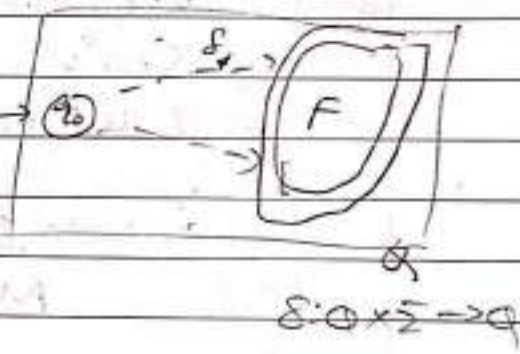
If we prove Reglang ⊆ CFL, (L → CFG)
then, Reglang − CFL = ϕ.



Assume:
Given a Regular language L,
∃ DFA M / lang(M) = L

(Ex)





N = {q0, q1, q2} (N, T, R, S)
T = Σ = {a, b}
S = q0
R:

$$\begin{cases} q_0 \to aq_1 & R_1 \\ q_0 \to bq_2 & R_2 \\ q_1 \to bq_1 & R_3 \\ q_1 \to aq_2 & R_4 \\ q_2 \to bq_2 & R_5 \\ q_2 \to aq_1 & R_6 \\ q_2 \to \in & R_7 \end{cases}$$

To prove lang(M) = lang(G_M)
(⇒) $w \in lang(M)$
$w = a_1 \ldots a_n$
$\Rightarrow \exists\ r(w) : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$

- DFA of multiple final states?
- minimal DFA → unamb?

DOMS | Page No.
Date / /

⇒ construct a derivation as:- $(S \Rightarrow^* w)$

$q_0 \rightarrow a_1 q_1 \rightarrow a_1 a_2 q_2 \rightarrow \dots \rightarrow a_1 \dots a_n q_n$

(q_0 — Start)

$a_1 \dots a_n$ (w)

$(q_n \rightarrow \epsilon)$

$(q_{n-1} \rightarrow a_n q_n)$

$\forall n \in [2, n]$

($\Leftarrow$) $w \in \text{lang}(G_M)$

$S \Rightarrow^* w$.

⇒ ∃ at least one derivation from q_0 to w $(q_0 \Rightarrow^* w)$

create DFA D',

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

$N = Q$

$S = q_0$

$T = \Sigma$

$R = \{ A \rightarrow aB \text{ if } \delta(A, a) = B \}$

$\cup$

$\{ A \rightarrow \epsilon \text{ if } A \in F \}$

2nd Part:

#steps = # non-leaf nodes

#leafnodes = $|w|$ (with some ϵ's)

$n + m \leq$ # nodes $\leq n + 2m - 1$

m right ϵ's

DOMS | Page No.
Date / /

→ Push Down Automata (Intro)

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

I/P tape

a	a	b	b	$

(Machine can only scan from left to right, returns if member or not)

finite control

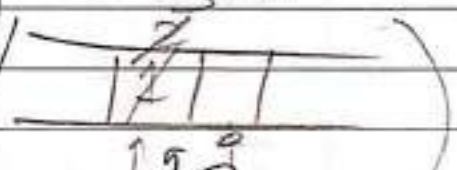
Start $\{q_0, \ldots, q_n\}$

Has the δ-fn

we need a memory tape, for the lang above (since it has infinite #eq.cl.)

↳ Roughwork/Scratchpad

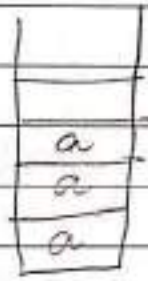
(1) (capabilities)



#a #b

(subtract from #a until zero)

(2) Stack (Auxiliary Memory)



(when b's come, pop until empty)

high level. we don't have this

(# mem = infinite...)

↳ no cap.

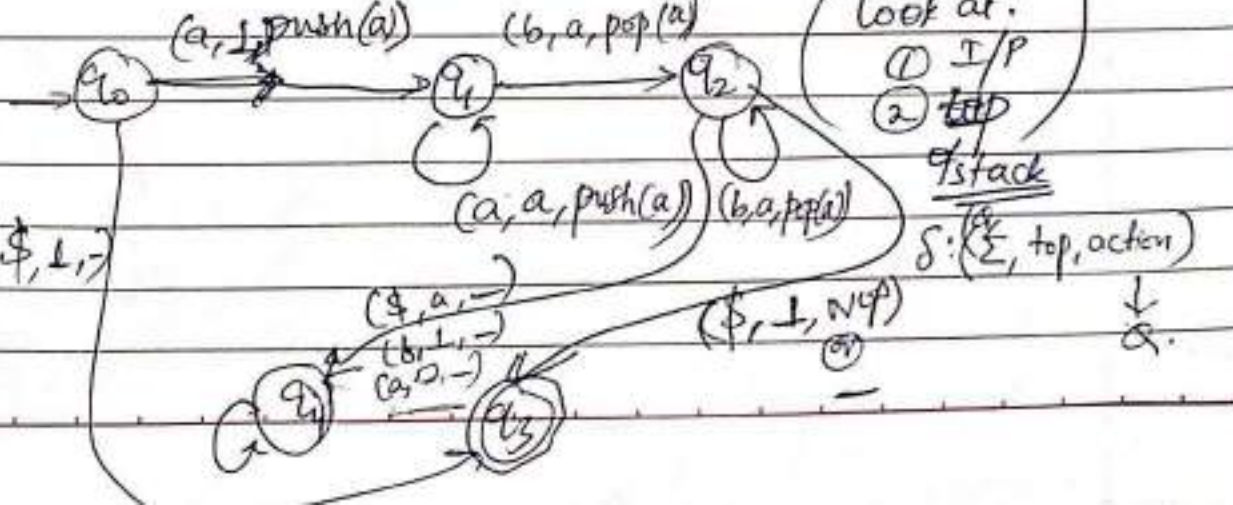
↳ even for finite alphabet

For finite word (obvious), it works (finite alphabet Regd)

conventions:

$ = end of word. ($\$ \notin \Sigma$)

⊥ = stack bottom. ($\perp \notin \Sigma$)



Look at:
① I/P
② top of stack

$\delta: (Q, \Sigma, \text{top}, \text{action}) \to Q$

DOMS | Page No.
Date 4/2/22

T-3.

1. $G = (N, \Sigma, R, S)$

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

$N = \{S, A\}$

$R = \{R_1, \ldots, R_5\}$

$R_1 : S \rightarrow AA$

$R_2 : A \rightarrow AAA$ ✗

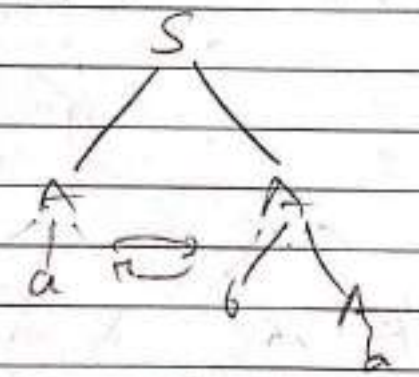
$R_3 : A \rightarrow a$

$R_4 : A \rightarrow bA$

$R_5 : A \rightarrow Ab$

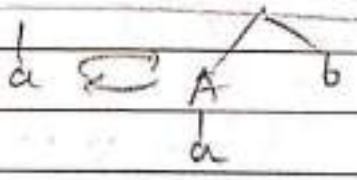
(1) Generate all strings w/ ≤ 4 steps in derivation

$R_1 R_3 R_3$



aba
aab
baa

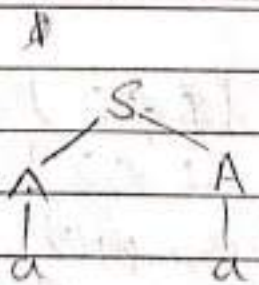
$R_1 R_3 R_4 R_3$



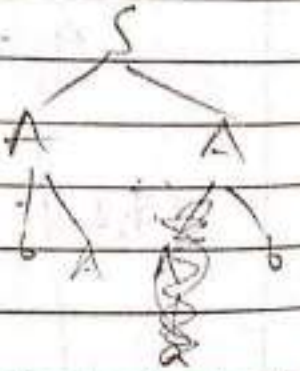
$R_1 R_4 R_3 R_3$

$R_1 R_5 R_3 R_4$

$R_1 R_3 R_3 R_4$



aa



DOMS | Page No.
Date / /

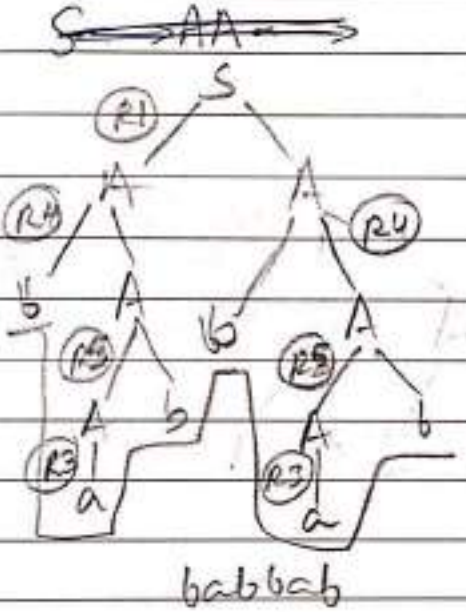
(2) ≥4 derivations for babbab.

$S \xrightarrow{R1} AA \xrightarrow{R4} bAA \xrightarrow{R4} bAbA \xrightarrow{R5} bAbbA \xrightarrow{R5} bAbbAb$

$\downarrow R3$

$\downarrow R3$

babbab



Derivations

① $R_1 R_4 R_4 R_5 R_5 R_3 R_3$

② $R_1 R_4 R_5 R_4 R_5 R_3 R_3$

③ $R_1 R_4 R_5 R_3 R_4 R_5 R_3$

④ $R_1 R_4 R_5 R_4 R_3 R_5 R_3$

⑤ $R_1 R_4 R_4 R_5 R_3 R_5 R_3$ ..

(3) For any $m, n, p \geq 0$, describe derivation of G in string $b^m a b^n a b^p$.

$S \xrightarrow[R1]{} AA \xrightarrow[R4]{(m\text{ times})} b^m AA \xrightarrow[R5]{(n\text{ times})} b^m A b^n A \xrightarrow[R5]{(P\text{ times})} b^m A b^n A b^p$

$\downarrow R3$

$\downarrow R3$

$b^m a b^n a b^p$

$\rightarrow$

$|\omega| + 1 [\Sigma]$

DOMS | Page No.
Date / /

2. Give CFG, PDA for : $\Sigma = \{a, b\}$

(a) $\{ww^R \mid w \in \Sigma^*\}$

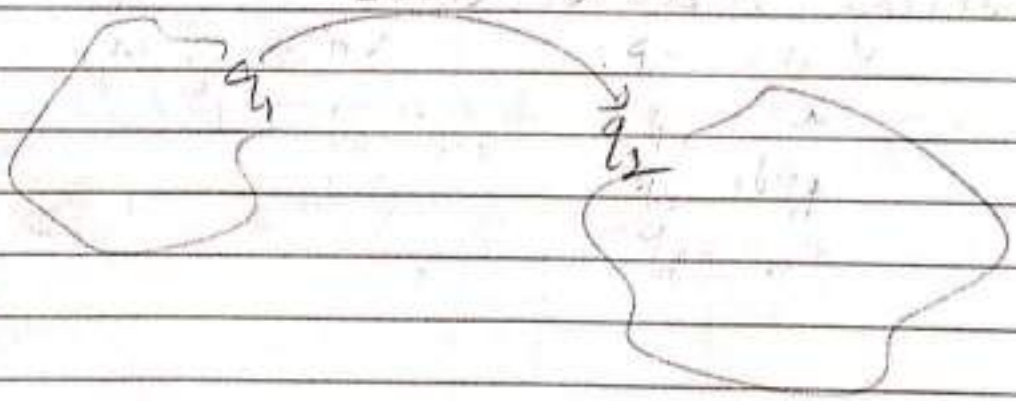
P_1: $S \to \epsilon$ (even no. palindrome)
P_2: $S \to aSa$
P_3: $S \to bSb$

$R = \{P_1, P_2, P_3\}$
$N = \{S\}$
$S = S$
$T = \Sigma = \{a, b\}$

No DPDA

(b) ~~[illegible]~~

Something. Just make a guess abt middle



DPDA << NDPDA
power-wise

(b) $\{wcw^R \mid w \in \Sigma^*\}$

$R_1 : S \to c$

$R_2 : S \to aSa$

$R_3 : S \to bSb$

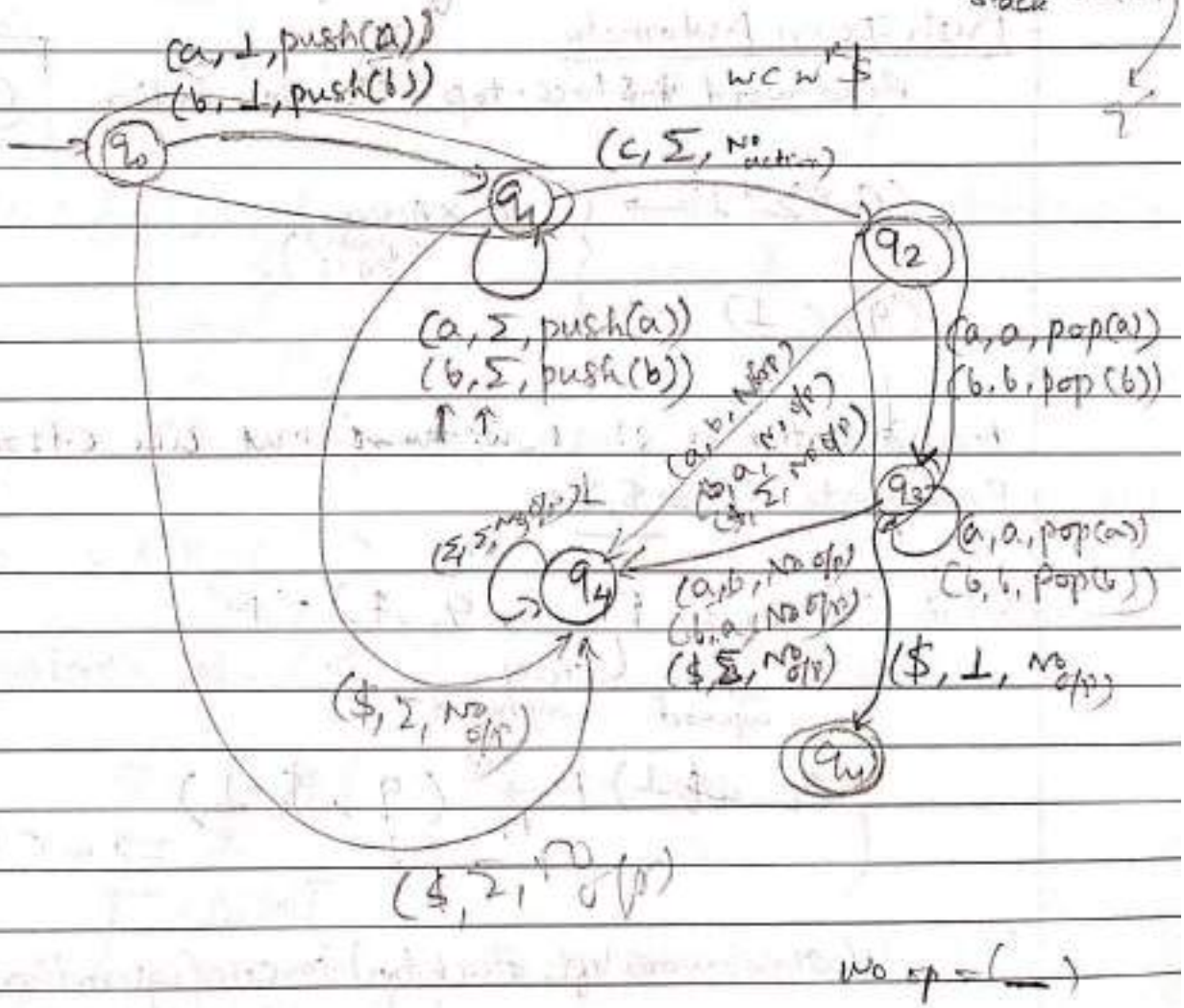
$R = \{R_1, R_2, R_3\}$

$N = \{S\}$

$T = \{a, b, c\} = \Sigma \cup \{c\}$

$S = S$

$\delta(q, \underbrace{\square}_{\text{alphabet}}, \underbrace{\square}_{\text{top of stack}}, \underbrace{\square}_{\text{action}})$



No o/p = (—)

DOMS | Page No.
Date 8/2/22

L-23

Outline
- CFGs, PDAs (generalize), CFG ⇒ PDA
- construction, example. Pf (⇒, ∈)

→ Recap
- Context Free languages
  $L \subseteq \Sigma^*$ is a CFL if there exists a CFG for it.
- Push Down Automata
  Read word + stack top, do some action.

$(Q \times \Sigma^* \times \Gamma) \rightarrow$ { $Q \times$ Action (Push() / Pop())

$(q, \epsilon, \perp) \rightarrow$ {

Manipulation of stack w/ no input like ∈-transitions is possible

Final state, if $\in \$, \perp$.

→ PDA : $(Q, \Sigma, \Gamma, \Delta, q_0, F) = M$
(Σ = I/P alphabet, Γ = stack alphabet)

$(q_0, w\$, \perp) \xrightarrow[M]{*} (q, \$, \perp)$, $q \in F$ $\Rightarrow w \in L(M)$

(state, word left, stack top) = configuration.

$(q, \$, \perp)$ = Accepting config.

If a run to an ~~accept~~ accepting config exists, then accept word. Otherwise, reject.
(Similar to NFA)

Let's generalize to ∈-NFA also.

DOMS | Page No.
Date / /

→ Goal: To show that PDA exists for any given CFL (i.e., CFG)

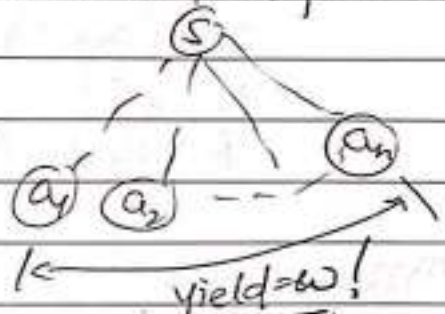
consider a CFG, $G = (N, \Sigma, R, S)$

(Goal: PDA, $M = (Q, \Sigma, \Gamma, \Delta, q_0, F)$)

A word $w \in$ lang(G) if it has atleast 1 derivation in it, that could be represented in parse trees.

$S \Rightarrow^* w$ (or)

General



Select leftmost derivation to avoid nondeterminism

[Sort of "preorder"]

$(S \Rightarrow_L^* w)$

More specific.

So $w \in$ lang(G) $\Rightarrow (S \Rightarrow_L^* w) \rightarrow$ (let's mimic this leftmost deriv^n in the PDA, M)

consider $M = (Q, \Sigma, \Gamma, \Delta, q_0, F)$, where,

$Q = \{p, q\}$ $\quad q_0 = \{p\}$

$\Sigma = \Sigma$ $\quad F = \{q\}$

$\Gamma = NUT$

Δ = transitions based on the rules R.

↳ 3 Types:

R: $A \to x$

① Initialize: $(p, \in, \perp) \to (q, \text{push}(S))$

② Replacement: $(q, \in, A) \to (q, x)$ "replace"

③ Matching & popping: $(q, a, a) \to (q, \text{pop}(a))$

DOMS | Page No.
Date / /

Ex) $L = \{wcw^R \mid w \in \Sigma^*\}$ $\Sigma = \{a, b\}$

$G = (N, \Sigma, R, S)$

$R_1: S \to c$
$R_2: S \to aSa$
$R_3: S \to bSb$
$\Rightarrow R_1': S \to c \mid aSa \mid bSb$ (Backus-Naur Form)

$S = S$
$\Sigma = \{a, b, c\}$
$N = \{S\}$
$R = \{R_1, R_2, R_3\}$ or $\{R_1'\}$

Transitions:

Type 1: $(p, \epsilon, \perp) \to (q, push(S))$

Type 3:
$(q, a, a) \to (q, pop(a))$
$(q, b, b) \to (q, pop(b))$
$(q, c, c) \to (q, pop(c))$

Type 2:
$(q, \epsilon, S) \to (q, c)$
$(q, \epsilon, S) \to (q, aSa)$
$(q, \epsilon, S) \to (q, bSb)$

old def^n: $(q, \epsilon, S) \to (q_i, \epsilon, pop(S)) \to (q, \epsilon, push(c))$

$(q, \epsilon, S) \to (q_i, \epsilon, pop(S)) \to (q_j, \epsilon, push(a))$ [a]
[S] ⊔ $(q_k, \epsilon, push(S))$ [Sa]
$(q_l, \epsilon, push(a))$ [aSa]

New def^n is just matter of convenience

consider: $w = abcba$
$w \in L(G)$
derivation: $S \Rightarrow aSa \Rightarrow abSba \Rightarrow abcba$

→

a b c b a $

(p)

⊥ ←

After Type ①,

(p) → (q)

a b c b a $

S ←
⊥

Using Type ②, (since Top = S) ∈ N

(p) → (q)

a b c b a $

a ←
S
a
⊥

Use Type ③ (since Top = a) ∈ T.

(p) → (q)

a b c b a $

S ←
a
⊥

Use type ②, (since Top = S) ∈ N

(p) → (q)

a b c b a $

b ←
S
b
a
⊥

Use type ③, (since Top = b) ∈ T

(p) → (q)

a b c b a $

S
b
a
⊥

Use type ② (since top = S) ∈ N

(p) → (q)

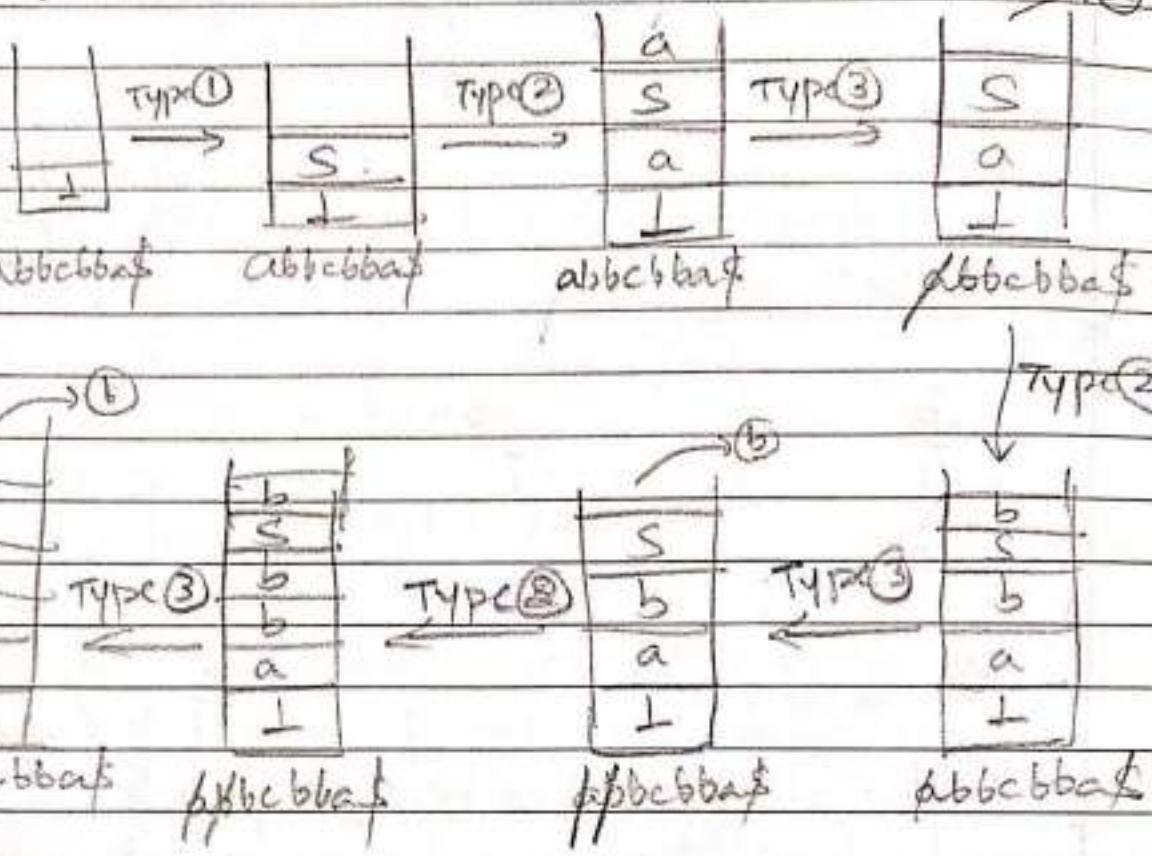
a b c b a $

c
b
a
⊥

Then, Type ③ thrice, to reach $, ⊥.

Soms | Page No.
Date / /

Sx) w = abbcbba



S ⊥	$(p, \epsilon, \perp) \xrightarrow{①} (q, push(S))$	
aSa ⊥	$(q, \epsilon, S) \xrightarrow{②} (q, aSa)$	
Sa ⊥	$(q, a, a) \xrightarrow{③} (q, pop(a))$	⊥ $(q, b, a) \xrightarrow{③} (q, pop(a))$
bSba ⊥	$(q, \epsilon, S) \xrightarrow{②} (q, bSb)$	↓
Sba ⊥	$(q, b, b) \xrightarrow{③} (q, pop(b))$	$(\$, \perp)$
bSba ⊥	$(q, \epsilon, S) \xrightarrow{②} (q, bSb)$	
Sba ⊥	$(q, b, b) \xrightarrow{③} (q, pop(b))$	
cba ⊥	$(q, \epsilon, S) \xrightarrow{②} (q, \epsilon)$	
cba ⊥	$(q, c, c) \xrightarrow{③} (q, pop(c))$	
ba ⊥	$(q, b, b) \xrightarrow{③} (q, pop(b))$	
a ⊥	$(q, b, b) \xrightarrow{③} (q, pop(b))$	

DOMS | Page No.
Date 9/2/22

L-24

Outline:

- CFG → PDA.
- Pf: (⇒), (⇐).
- PDA → CFG., pf, examples.

→ Go to L25

During CFG-to-PDA conversion,

$G = (N, \Sigma, R, S)$

$M = (Q, \Sigma, \Gamma, \Delta, \delta, F)$

where Δ:
- $(p, \epsilon, \perp) \to (q, \text{push}(S))$
- $(q, \epsilon, s) \to (q, aSa)$
- $(q, a, a) \to (q, \text{pop}(a))$

Initialize:
$Q = \{p, q\}$, $\Sigma = \Sigma$, $\Gamma = NUT = N \cup \Sigma$, $s = p$, $F = \{q\}$

→ Replacement and i/p processing (matching)

This PDA mimics $S \Rightarrow_L^* w$

Is this algo. correct? (i.e. Is $L(G) = L(M)$?)

To prove $w \in L(G) \Leftrightarrow w \in L(M)$

$w \in L(G) \Leftrightarrow (S \Rightarrow_L^* w)$

$\Downarrow$

$(p, w, \perp) \vdash_M^* (q, \$, \perp)$

$w \in L(M) \Leftrightarrow$

If w is in language of G, it has a leftmost derivation from S.

Our pf. needs to establish:

$(S \Rightarrow_L^* w) \Leftrightarrow ((p, w, \perp) \vdash_M^* (q, \$, \perp))$

Consider claim: $S \Rightarrow_L^* wx$ (iff), $w \in \Sigma^*$, $\alpha \in (N)(NUT)^*$

$(q, w, s) \vdash^* (q, \alpha)$

derived; still on stack (First char of α is top of stack)

If α was somehow $\in$ in our claim, it looks similar to the goal pf.

$(S \Rightarrow_L^* w) \iff (q, w\$, S) \vdash_M^* (q, \$, \perp)$

[we removed first implicit transition $(p, w\$, \perp) \to (q, w\$, S)$]

which is similar to

$(S \Rightarrow_L^* w\alpha) \iff (q, w, S) \vdash_M^* (q, \epsilon, \alpha)$ [with $\alpha = S$]

If $\alpha \neq \epsilon$

($\Rightarrow$) Assume $S \Rightarrow_L^* w$.

→ Induction on #steps in leftmost derivation of w.

Base case: #steps = 0

$S \to *$

$S \to S$

Bad base case

Base case: #steps = 1

$S \Rightarrow x\,A\,\alpha$ ($w \in N$)

(or) $S \to a$ (i.e. $S \to w$.) ($w = a$)

$(p, w\$, \perp) \vdash (q, w\$, S) \to (q, w\$, \alpha)$

Assumed implicit

(Imp) $(q, \$, \perp)$

NOTE: one rule applic^n in grammar ⇒ one Type ② Transition in PDA

DOMS | Page No.
Date / /

Configuration: $w = xy$, $|x| = n$.
When we have finished deriving 'x', at that moment, 'α' is on the stack.
(If $y \neq \epsilon$, $\nexists \beta \Rightarrow \phi \perp$)

i.e.,

$S \Rightarrow_L^{n\text{-steps}} xA\beta = w\alpha$ (Un)

$\alpha = A\beta$; $(n+1)^{th}$ step ; $(A \to \gamma)$ in R

$w\gamma\beta$ (Un+1)

By induction hypothesis,

$(q, w, S) \vdash_M^* (q, \epsilon, \alpha) = (q, \epsilon, A\beta)$ — Type ② → $(q, \epsilon, \gamma\beta)$

| A |
| β |

→

| γ |
| β |

In first 'n' steps,

Run of machine: $(q, w, S) \vdash^* (q, y, \alpha) \dashrightarrow (q, \$, \gamma\beta)$; $(n+1)^{th}$ step.

($\alpha = A\beta$)

part of w | Type ② Transition

$S \to x\alpha$; $\alpha \to A(\Sigma \cup V)^*$

| A |
| β |

$= xy$

| γ |
| β |

$= y$

(?)

$(q, \$, \perp)$

DOMS | Page No.
Date / /

Goal: $S \Rightarrow_L^* w \longrightarrow (q, \$, \perp)$

$w = xy$

Given deriv^n: $\left(S \Rightarrow_L^* \alpha A \beta \Rightarrow \alpha\gamma\beta \Rightarrow_L^* xy = w \right)$

(n steps; $(n+1)^{th}$ step.)

If $\gamma\beta \Rightarrow_L^* y$, then $(q, y, \gamma\beta) \overset{?}{\Rightarrow} (q, \$, \perp)$

from deriv^n, $\boxed{\gamma\beta \Rightarrow_L^* y}$

If $\gamma\beta \xrightarrow{\leq n \text{ steps}} y$, then $(q, y, \gamma\beta) \xrightarrow{\text{By Ind. Hyp.}} (q, \$, \perp)$

~~If ind hyp — w is processed~~

~~What is~~ Induction Process

Goal: $S \Rightarrow_L^* w \iff (q, w\$, S) \vdash^* (q, \$, \perp)$

Base: 1 step: $S \to a$, $(w = a)$

So,

$(q, a\$, S) \vdash (q, \$, a) \longrightarrow (q, \$, \perp)$

Ind. Hyp: n-steps: $S \to \alpha \to \alpha_1 \to \cdots \to_L^* w$ (n-steps)

DOMS | Page No.
Date 11/2/22

L-25

outline
- CFG → PDA, pf.
- PDA → CFG

→ (iff) G = (N, Σ, R, S)
(iff) M = (Q, Σ, Γ, δ, q0, F) Q = {p, q} F = {q}
q0 = p

① (p, ∈, ⊥) → (q, push(S))
② (q, ∈, A) → (q, push(γ)), if A → γ ∈ R
③ (q, a, a) → (q, pop(a)), ∀a ∈ Σ

Theorem: L(G) = L(M). {pf. of correctness of our algorithm}

(To prove) [∀w. (S ⇒* w iff (q, w$, S) ⊢* (q, $, ⊥))]
[((p, ∈, ⊥) → (q, push(S))) → Implicit]

(⇒) Induction on #steps in leftmost derivation of w.

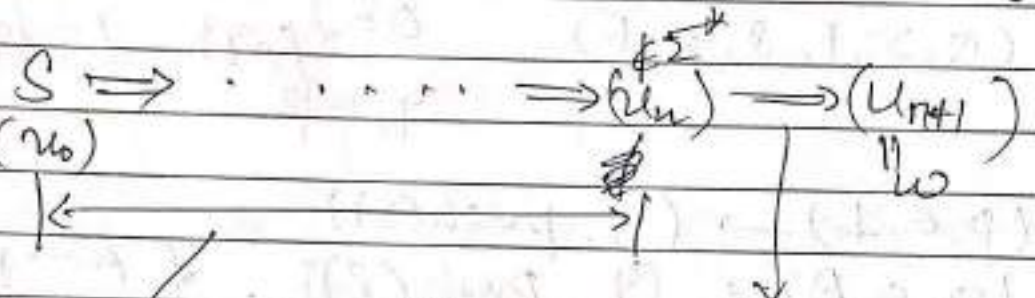
Base case:
P(0): #steps = 0 : S → X (∅) → can't be used
P(1): #steps = 1 : S → a (or) ∈, a ∈ Σ*
(q, a$, S) ⊢ (q, a$, a) Type ②
⊢ (q, $, ⊥) Type ③
(q, ∈$, S) ⊢ (q, $, ⊥) Type ③

P(n): #steps = n: [S ⇒ⁿ w (w ∈ Σ*)
⇓
(q, w$, S⊥) ⊢* (q, $, ⊥)]
(Ind. Hyp.)
(Proof of)

(old) Inductive step : (uhhh)

$P(n+1)$: $\left(S \Rightarrow^{n+1} w\right)$ # Prove this using P(n)

To prove: $\left[(q, w\$, S) \vdash^{*} (q, \$, \perp)\right]$



Can't apply Ind. Hyp!!

If this is $A \to \gamma$, then u_n should have a non-terminal.

I.e: $u_n = \alpha A \beta$.

$u_{n+1} = \alpha \gamma \beta$.

Let's modify induction hypothesis to make it useful form.

Intuitively,

→

DOMS | Page No.
Date / /

Induction Hypothesis: $w\alpha \in (NUT)^*$

$$p(n): \left[(S \Rightarrow^{L^n} w\alpha) \Rightarrow (q, w\$, S) \vdash^* (q, \$, \alpha) \right]$$

Assuming I.H., can we prove $(n+1)^{th}$ step?

~~$\alpha = \epsilon$~~ in $p(n)$ gives us ~~[illegible]~~ our goal statement.

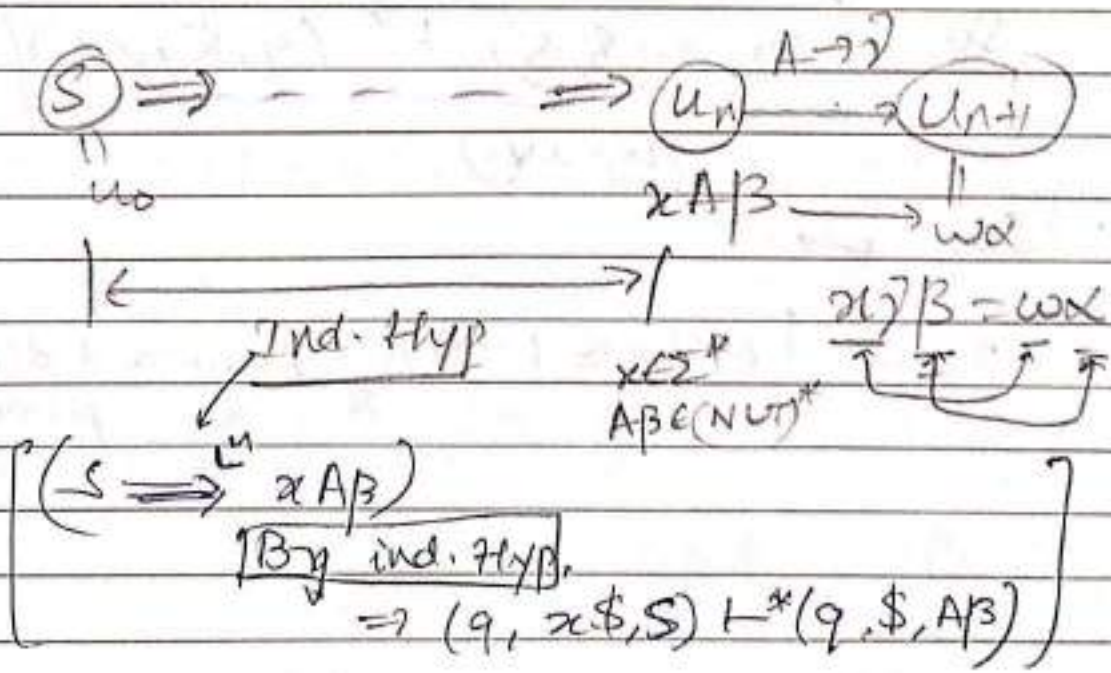
~~Inductive Step:~~

~~$p(n+1): S \Rightarrow^{L^{n+1}} w\alpha$~~

Base case:

For $p(0)$, $S = w\alpha$, forces $\begin{pmatrix} w=\epsilon \\ \alpha = S \end{pmatrix}$ which is trivial

To prove $\left((S \Rightarrow^{L^{n+1}} w\alpha) \Rightarrow (q, w\$, S) \vdash^* (q, \$, \alpha) \right)$



$$\left[(S \Rightarrow^{L^n} \chi A\beta) \text{ By ind. Hyp.} \Rightarrow (q, x\$, S) \vdash^* (q, \$, A\beta) \right]$$

So, $(q, x\$, S) \vdash^* (q, \$, A\beta) \vdash (q, \$, \gamma\beta)$

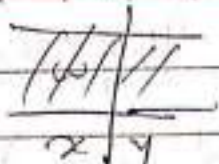
Type ②. corresponding to $A \to \gamma \in R$

Note: x is portion of w that has been processed in first n-steps

DOMS | Page No.
Date / /

$w\alpha = xy\alpha$
$= x\gamma\beta$

$\Rightarrow y\alpha = \gamma\beta$



$w = xy$

Some prefix of $\gamma\beta$ is 'y', Rest is α.

For $w = xy$,

$(n+1)^{th}$ step.

$(q, xy\$, S) \vdash^* (q, y\$, A\beta) \vdash (q, y\$, \gamma\beta)$

$(y \in \Sigma^*)$

~~we can get~~ y

we can pop prefix of $\gamma\beta$ corresponding to y only using type ③ transitions. Only α is left on stack.

So, $\boxed{(q, xy\$, S) \vdash^* (q, \$, \alpha\perp)}$

$(w = xy)$

Hence, $\boxed{L(G) \subseteq L(M)}$ forward dir^n proved

$\rightarrow$

DOMS | Page No.
Date / /

($\Leftarrow$) To prove backward direction:

$$\left((q, w\$, S) \vdash^* (q, \$, \perp)\right)_{\text{PDA}} \Rightarrow \left(S \Rightarrow^{L^*} w\right)$$

via Induction of #type2-transitions in the run.

To prove: $P(n)$: $(S \Rightarrow^* w\alpha) \in (N \cup T)^*$ } Ind. Hyp

$\left((q, w\$, S) \vdash_M^n (q, \$, \alpha\perp)\right)$

Since each run of machine is type 2 transition
$\Updownarrow$
(one rule applic^n in grammar = one Type② Transition in PDA)

Base case:
$P(0)$: $\longrightarrow \begin{pmatrix} w=\epsilon \\ \alpha=S \end{pmatrix}$ which is trivial.

To prove:
$P(n+1)$: $\left[(q, w\$, S\perp) \vdash^{n+1} (q, \$, \alpha\perp) \Rightarrow (S \Rightarrow^{L^*} w\alpha)\right]$

$(q, w\$, S\$) \vdash^* (q, y\$, A\beta) \vdash (q, y\$, \alpha)$ ($A \to \gamma$). α/β

$\because$ [Type2 : $(q, \epsilon, A) \to (q, \gamma)$]

$w = xy$, x was processed in first 'n' type2-transition steps.

After $A \to \gamma$, using Type③ Transitions, $\vdash^* (q, \$, \alpha\perp)$.

By ind. hyp.

$(q, w\$, S1) \vdash^* (q, y\$, A\beta)$

Ind Hyp.

$(q, x\$, S1) \vdash^* (q, \$, A\beta)$

So, $S \Rightarrow^* xA\beta$ — Type ② → $x\gamma\beta$ — Type ③ → $xy\alpha$

$\omega\alpha$ || $(y\alpha = \gamma\beta)$

Hence $S \Rightarrow^* \omega\alpha$

So, $\boxed{S \Rightarrow^* \omega\alpha}$ $\quad \omega\alpha \in (N \cup T)^*$.

Hence proved,

$L(M) \subseteq L(G)$

Thus, $\boxed{L(M) = L(G)}$

Hence, our algo for CFG → PDA has been proved.

DOMS | Page No.
Date 15/2/22

L-26

outline
- PDA → CFG
- Examples — L(a^n b^n).

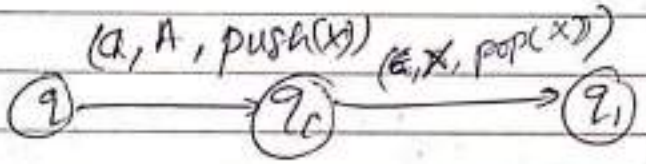
→ Simple PDA: (only push, pop)

PDA → Simple PDA

(a, A, pop(A))

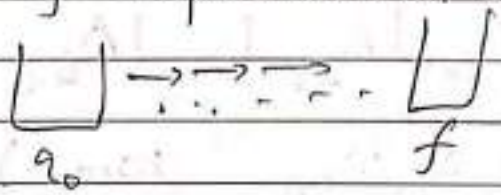
(a, A, no-op) ✗ Don't allow.

push some dummy symbol then pop it.

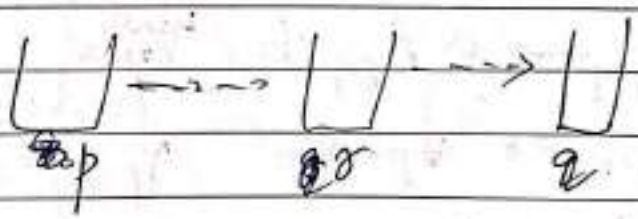


Consider a simple PDA, $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$.
We want a CFG, $G = (N, \Sigma, R, S)$, such that $L(M) = L(G)$

Examine any computation on $w \in L(M)$

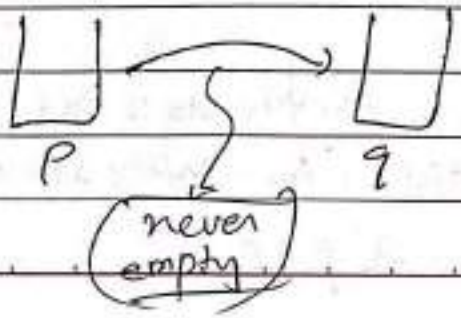


Case ①

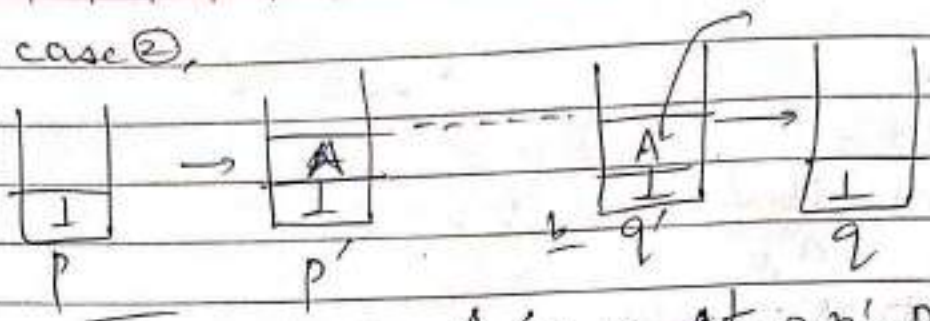


multiple encounters w/ empty stack are handled inductively

Case ②



In case ②,



$(p, a, A^{\perp} \to p', push(A))$

$(q', b, A \to q, pop(A))$

$\forall p, p', q, q' \in Q$

Grammar construction:

$G = (N, \Sigma, R, S)$

$S = S$

$\Sigma = \Sigma$

$N = A_{p,q}$ → M processes some part of lang from p to q. that string is $A_{p,q}$ (part of w)

Satisfying words: $A_{q_0,f}$ for all $f \in F$ → All strings: $q_0 \to f$.

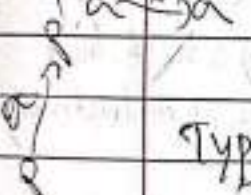
R:

$S \to A_{q_0,f_1} | A_{q_0,f_2} | \dots | A_{q_0,f_n}$, $|F| = |\{f_1, \dots, f_n\}| \geq 1$

$A_{p,q} \to A_{pr} \cdot A_{rq}$ (Handle case ①) — General concatenation → $\forall p, q, r \in Q$

$A_{p,q} \to a \cdot A_{p'q'} \cdot b$ (Handle case ②) $| a, b \in \Sigma \cup \{\in\}$ $(\forall p, q, p', q' \in Q)$

$A_{p,p} \to \in$ $\forall p \in Q$



Types of Rules in R:

T① $S \to A_{q_0,f}$

T② concatenation (empty stack in b/w)

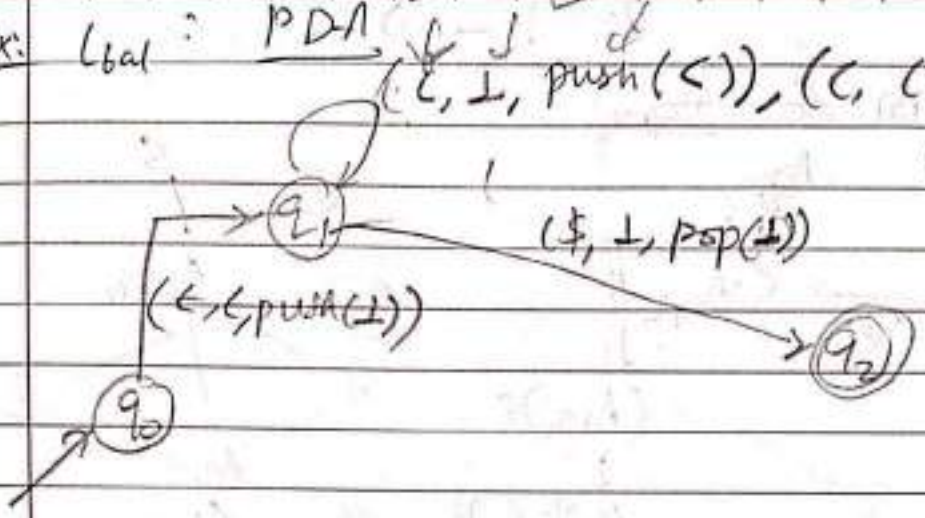
T③ Rule application (No empty stack in b/w)

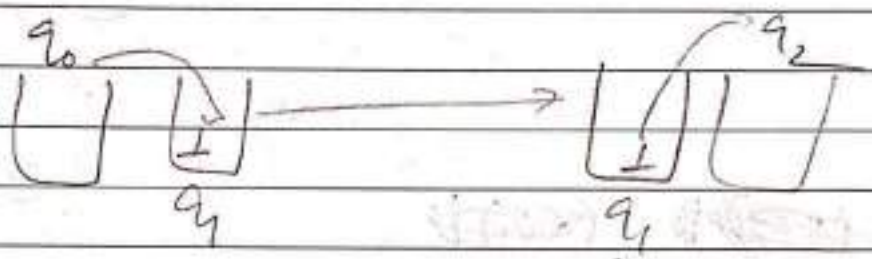
T④ $A_{pp} \to \in$ $\forall p \in Q$

$\delta(q, \underset{w}{\square}, \underset{t}{\square}) \rightarrow (q', \square)$

w | op

Ex: L_{bal} : PDA





CFG

Rules: $S \rightarrow A_{q_0,q_2}$

$N = \{S, A_{p,q}\}_{\forall p,q \in Q}$

~~$A_{q_0,q_2} \rightarrow A_{q_0,q_1} A_{q_1,q_2}$~~

$A_{q_0,q_2} \rightarrow (A_{q_1,q_1}) A_{q_1,q_2}$ (1) (2)

$A_{q_1,q_1} \rightarrow A_{q_1,q_1} A_{q_1,q_1} \mid A_{q_1,q_0} A_{q_0,q_1} \mid A_{q_1,q_2} A_{q_2,q_1}$

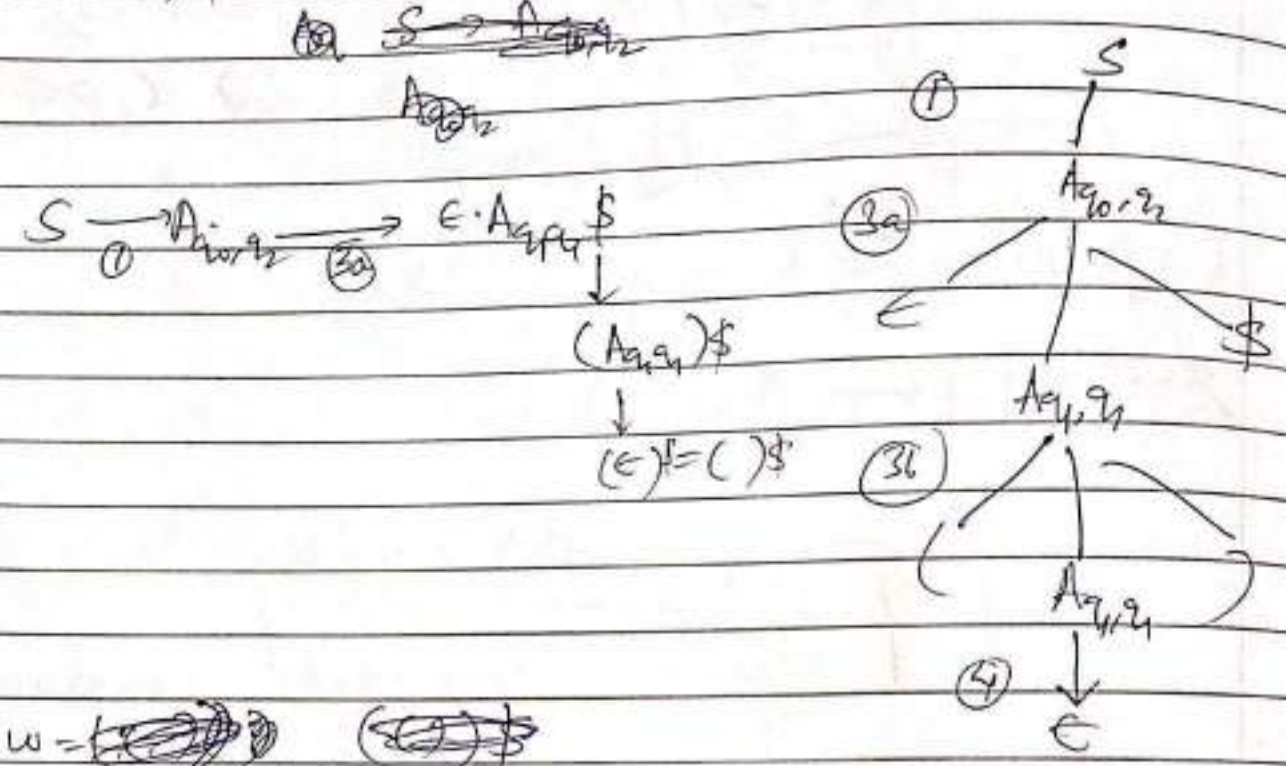
ϕ ϕ

$A_{q_1,q_1} \rightarrow (A_{q_1,q_1}) \mid \epsilon$

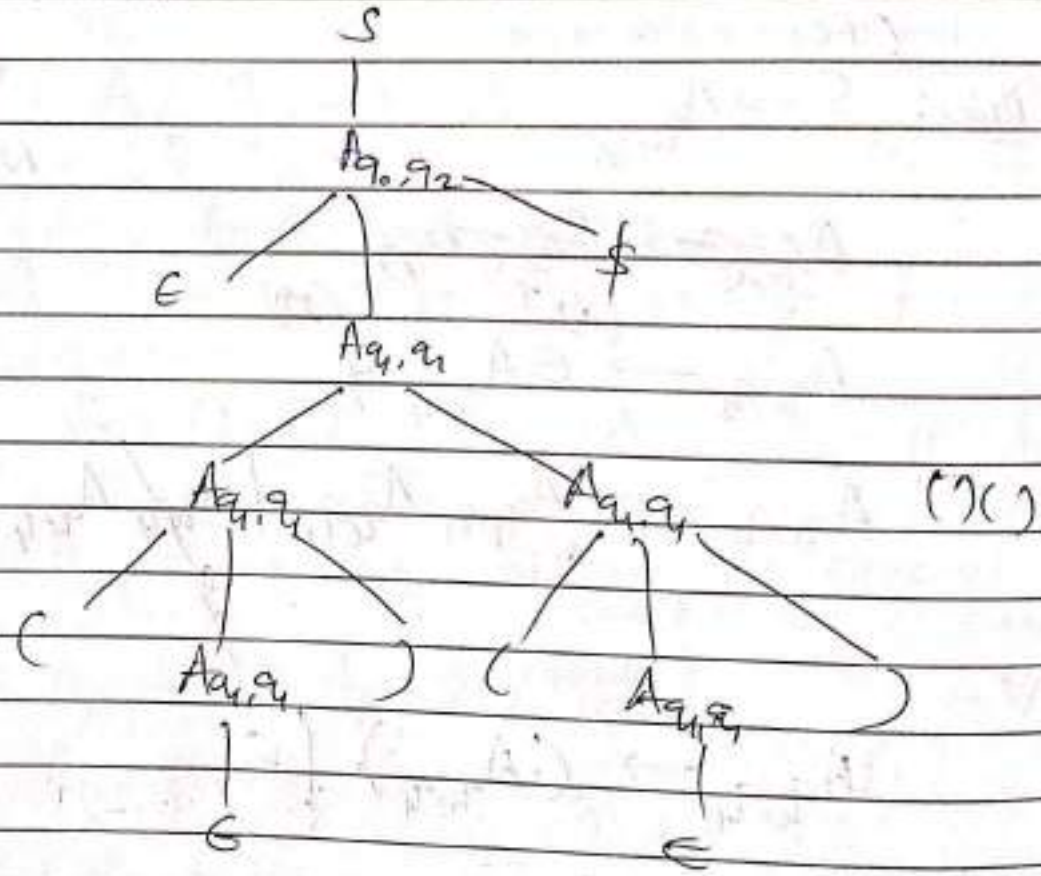
$A_{q_0,q_0} \rightarrow \epsilon$

$A_{q_2,q_2} \rightarrow \epsilon$

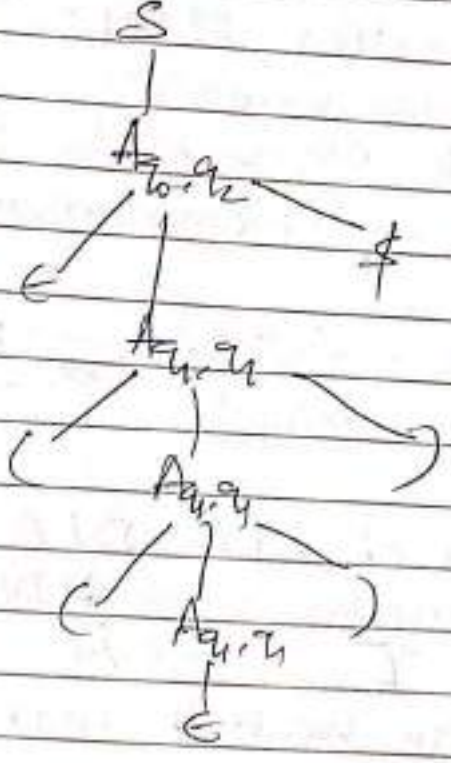
$w = ( )\$$:


S
A_{q_0,q_2}
$\epsilon \cdot A_{q_1,q_1} \$$
$(A_{q_1,q_1})\$$
$(\epsilon)\$ = ( )\$$
ϵ
$\$$
A_{q_1,q_1}
A_{q_1,q_1}
ϵ

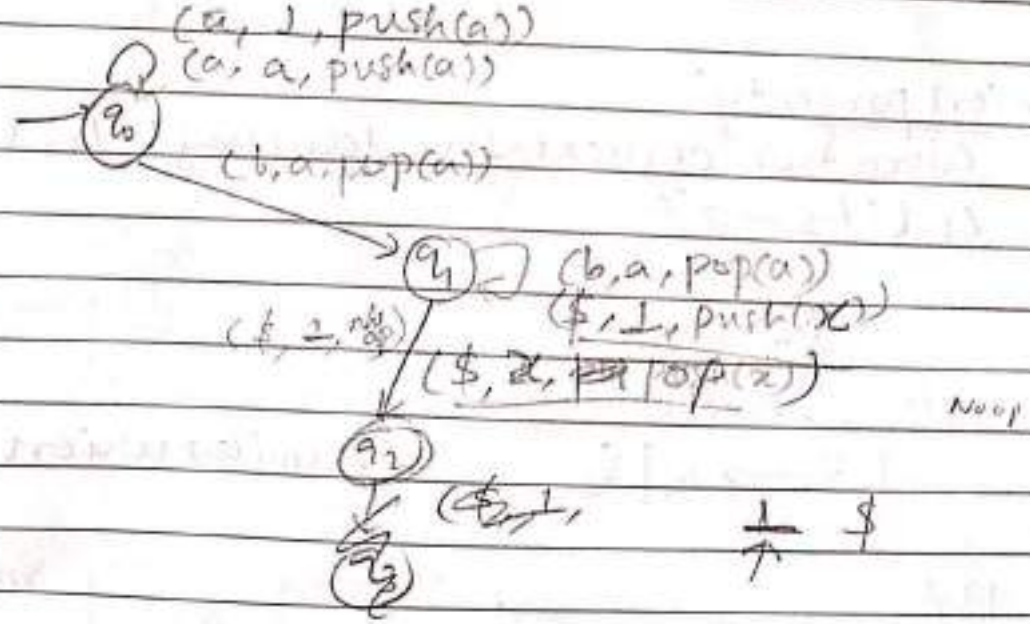
$w = (\,)(\,)\$$

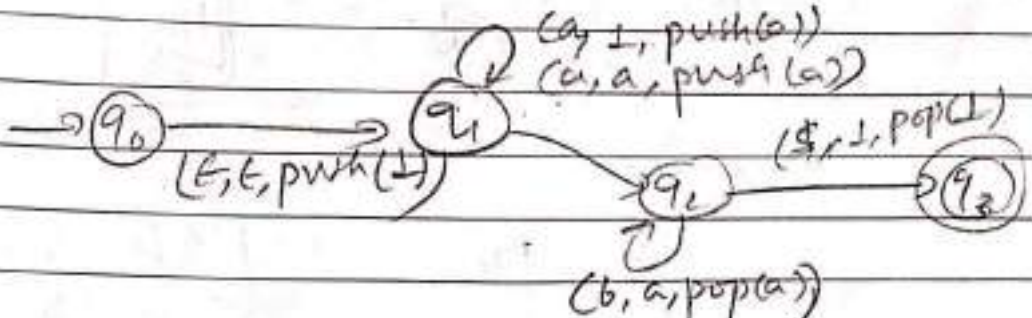

S
A_{q_0,q_2}
ϵ
$\$$
A_{q_1,q_1}
A_{q_1,q_1}
A_{q_1,q_1}
A_{q_1,q_1}
A_{q_1,q_1}
ϵ
ϵ
$(\,)(\,)$

Ex $w = (())\$$



Ex $a^n b^n$: PDA.





PDA → CFG
PS: Assgn

DOMS | Page No.
Date / /

L-27 Outline.

- closure properties of CFLs : union, concat, Kleene star
- pumping lemma (Ex. $a^n b^n c^n$)
- closure under intersection
- closure under complementation

→ CFLs → CFGs → (N)PDA ⇒ DPDA

Ex) ww^R Ex) $w^* c w^R$

→ Determination of NFA → DFA.

when L is regular ⇝ DFA M'

$\bar{L}$ ⇝ $\bar{M}$

But, complementation wouldn't work for NFAs.

→ Since DPDA << NPDA, we can't "guarantee" closure (yet).

→ Union property:

Given two context-free languages L_1, L_2.

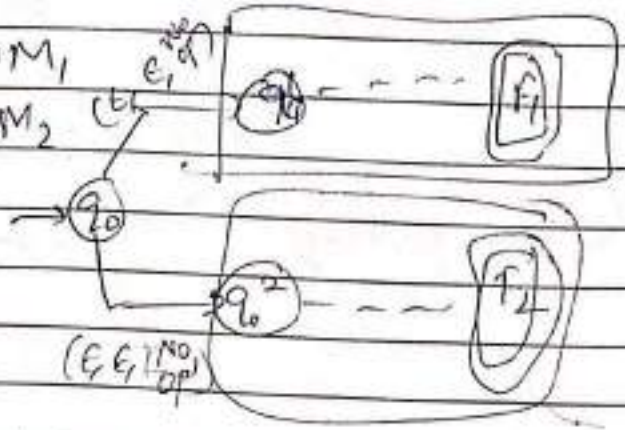
$L_1 \cup L_2 \rightarrow ?$

CFG
- $S \rightarrow L_1 \cdots$
- $S_2 \rightarrow L_2 \cdots$
- $S \rightarrow S_1 | S_2$ // handles union

PDA:

$L_1 \rightarrow M_1$

$L_2 \rightarrow M_2$



DOMS | Page No.
Date / /

$G_1 = (N_1, \Sigma_1, R_1, S_1)$
$G_2 = (N_2, \Sigma_2, R_2, S_2)$ $\left(\begin{matrix} S_1 \Rightarrow^* \\ S_2 \Rightarrow^* \end{matrix}\right)$

$G = (N_1 \cup N_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{R'\}, S)$

$R' = (S \rightarrow S_1 | S_2)$

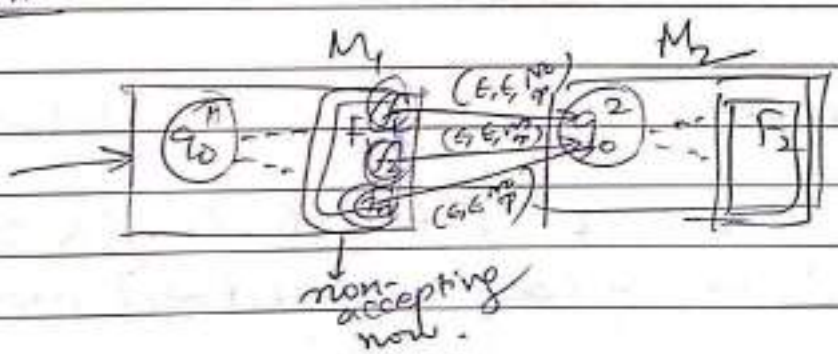
→ concatenation property:

$G_1 = (N_1, \Sigma_1, R_1, S_1)$
$G_2 = (N_2, \Sigma_2, R_2, S_2)$ $\left(\begin{matrix} S_1 \Rightarrow^* \\ S_2 \Rightarrow^* \end{matrix}\right)$

$G = (N_1 \cup N_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{R'\}, S)$

$R' = (S \rightarrow S_1 \cdot S_2)$

PDA:

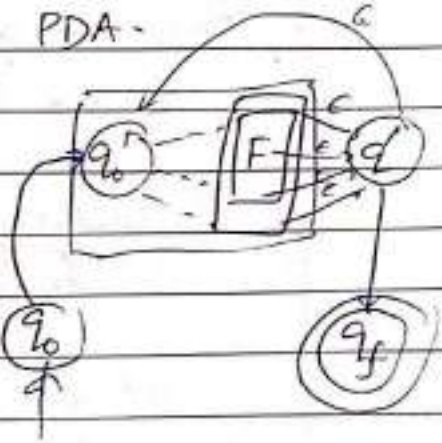


→ Kleene Star property

$G_1 = (N_1, \Sigma_1, R_1, S_1)$
$G = (N_1 \cup \{S\}, \Sigma_1, R_1 \cup \{R', R''\}, S)$

$R': S \rightarrow \epsilon$
$R'': S \rightarrow S_1 \cdot S$
$R: (S \rightarrow S_1 S | \epsilon)$

PDA:



DOMS | Page No.
Date / /

→ pumping lemma for CFLs : Intro

Ex) Try to construct NPDA for $L' = \{a^n b^n c^n \mid n \geq 0\}$
2-time memory retrieval

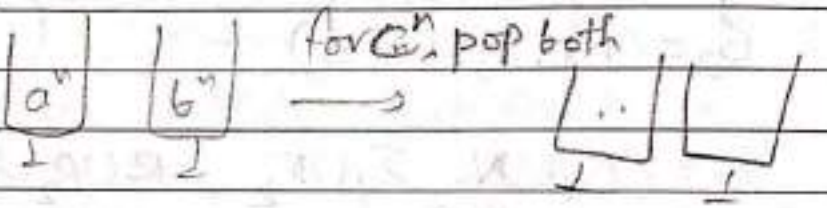
Idea for pumping lemma:

Assume ∃ NPDA "M" = $(Q, \Sigma, \Gamma, \delta, q_0, F)$

$|Q| = n$

$w = a^n b^n c^n$

With 2 stacks: $\delta : Q \times \Sigma \times \Gamma_1 \times \Gamma_2 \longrightarrow Q \times$ Actions

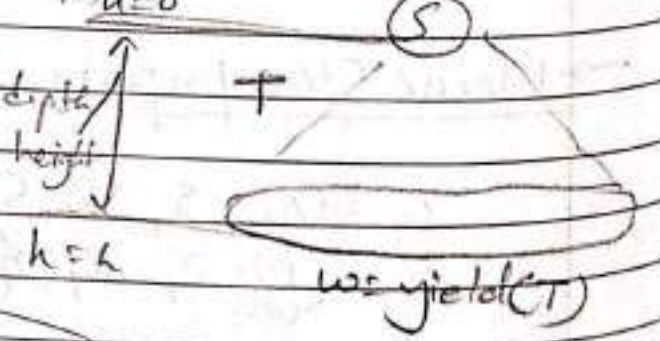


($\$, \perp, \perp$, noop)

Increasing no. of stacks also increases ↓ power of the machine.

~~Pumping lemma~~

→ Consider a CFL, 'L', with CFG, 'G' = (N, Σ, R, S), $w \in L(G)$, i.e., 'w' has a parse tree 'T' in G.



How many nodes in parse tree of height 'h'.

max Branching factor / Fanout : $\phi(G)$ → max. no. of symbols on RHS of any $\in R$

↳ max. branching factor

DOMS | Page No. | Date / /

Max Nodes per level: $[\phi(G)]^h$

$$\text{yield}(T) \leq (\phi(G))^h$$

$$|w| \leq \phi(G)^h$$

→ **Pumping lemma for CFLs:**

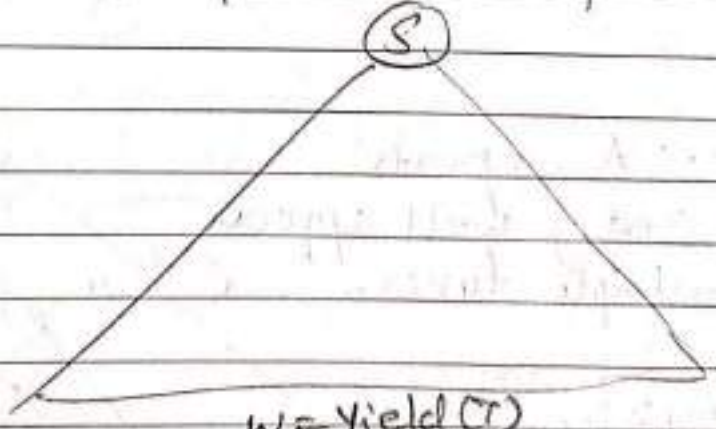
For a CFL 'L' with CFG 'G', consider $w \in L$, such that $w = uvxyz$, $vy \neq \epsilon$ ($|w| \geq \phi(G)^k$)
$\Rightarrow w' \in L$ such that,
$\Rightarrow w' = uv^n xy^n z \; \forall n \geq 0$

Choosing word:

$$\begin{pmatrix} w = uvxyz \\ vy \neq \epsilon \\ |w| \geq \phi(G)^{|N|} \end{pmatrix}$$

If $w \in L$, $\Rightarrow$ for a CFL, $w' = uv^n xy^n z \in L \; \forall n \geq 0$

Pf: construct a parse tree T for w in L.



$\text{yield}(T) \leq (\phi(G))^h$ → $|N|$ is no. of non terminals

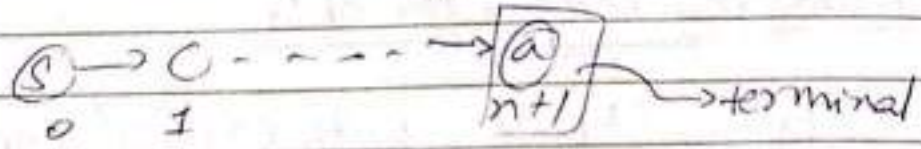
$|w| \geq \phi(G)^{|N|}$ → given by statement of pumping lemma

DOMS | Page No.
Date / /

$\phi(G)^{|N|} < |W| = |yield(T)| \leq \phi(G)^h$

$\Rightarrow \boxed{h \geq |N|+1}$

$\Rightarrow \exists$ a node 'a' at height $|N|+1$



$\Rightarrow \boxed{N+2}$ states in reaching leaf 'a'.

$\Rightarrow \boxed{N+1}$ Non-terminals on the way

$\Rightarrow$ ~~Sol~~ Atleast one non-terminal appears atleast twice

By PHP

$\Rightarrow$ we can re-invoke the set of rules that repeat ϕ (w.r.t. those repeating non-terminals)

If $A \rightarrow \cdots \Rightarrow A$ repeats, then v and y will appear multiple times.,

as:

$uvxyz \in L$ ← word

$\downarrow$

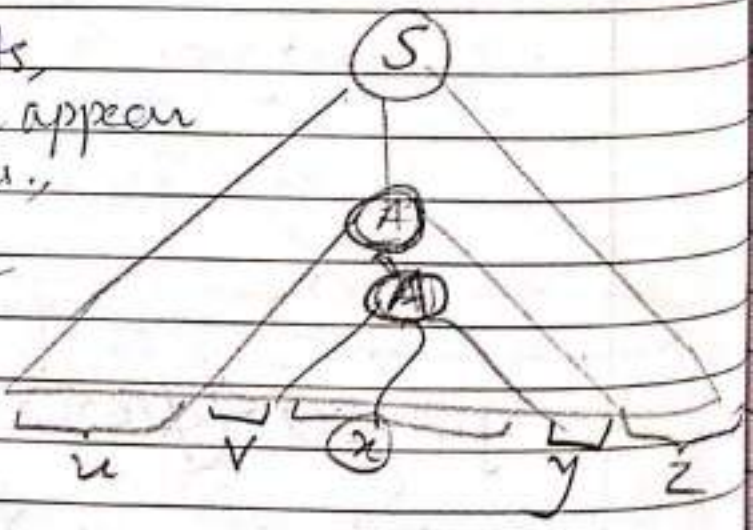
$uv^2xy^2z \in L$

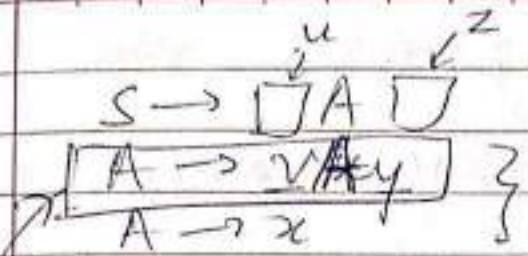
$\downarrow$

$\vdots$

$\downarrow$

$uv^nxy^nz \in L$





This is where the pumping occurs!

$uAz \Rightarrow^{k} uvAyz \Rightarrow uvxyz$

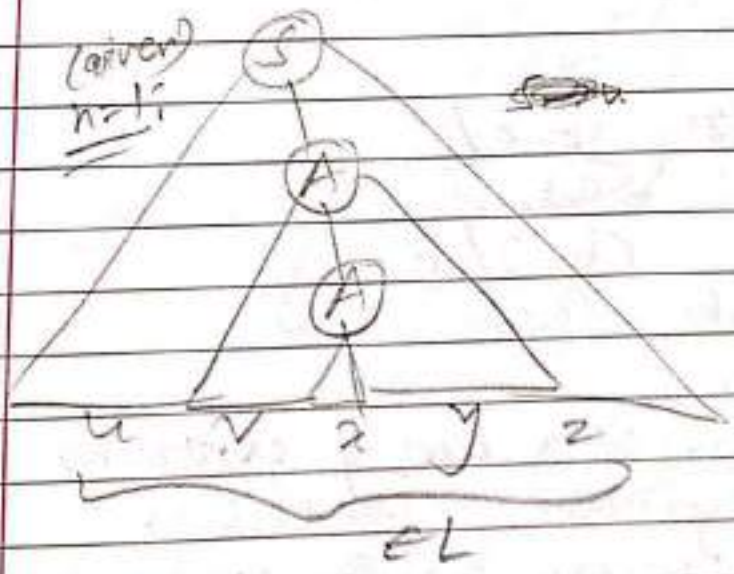
$\Rightarrow uv^2Ay^2z \Rightarrow uv^2xy^2z$

etc

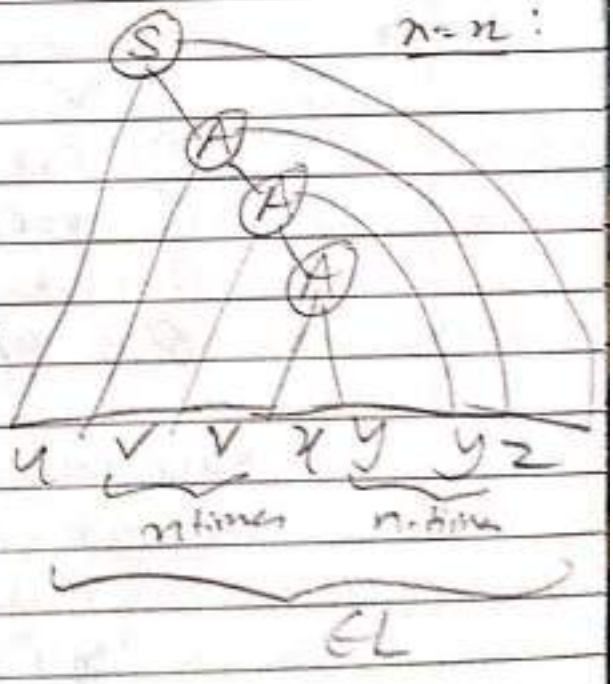
$\Rightarrow uv^nAy^nz \Rightarrow uv^nxy^nz$

Similar for as:

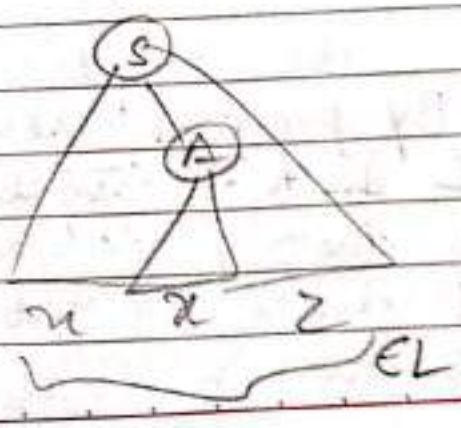
(proved) n=1:



n=n:



n=0:



If $vy = \epsilon$,
there's nothing to pump!

DOMS | Page No.
Date / /

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

Assume L is a CFL. So, it has a CFG.
Let us pick some 'N'
then some word w, such that $|w| > \phi(G)^{|V|}$
Let this word be,

$$w = a^n b^n c^n \quad | \quad n > \frac{\phi(G)^{|V|}}{3}$$

Then, we must decompose w into uvxyz, $|vy| \neq \epsilon$.

$w = a \ldots a\, b \ldots b\, c \ldots c \rightarrow (uvxyz)$.

case① → v, y may have a, b, c in them.
Assume $vy = a^{l_1} b^{l_2} c^{l_3} \mid l_1, l_2, l_3 > 0$.

	v	y
①	(a, b, c)	$v = \epsilon$ / (c)
②	$v = \epsilon$ / (a)	(a, b, c)
③	(a, b)	(b, c) / (c)
④	a / (a, b)	(b, c)

In each case atleast one of v and y contains at least 2 symbols.

(Now, we must pump v, y to produce a word $w = uv^n xy^n z \notin L$)

let $w' = uv^2xy^2z$ (By pumping lemma, $w' \in L$)
In case①, $w' \notin L$ due to 'v' (abc)
In case②, $w' \notin L$ due to 'y' (abc)
In case③, $w' \notin L$ due to 'v' (aabbbb)
In case④, $w' \notin L$ due to 'y' (bcbc)

Hence for all cases of u, v, x, y, z, $vy \notin L$, $w' \notin L$.

This is a contradiction to pumping lemma of CFL.

$\Rightarrow$ our assumption was incorrect, L is not a CFL!

Hence proved for case 1.

Case ② → v, y use only 2 symbols (a,b) or (b,c)

	v	y	
①	(a,b)	(b/c)	for (a,b); similarly, (b,c)
②	(a/b)	(a,b)	

Let $w' = uv^2xy^2z$. (By pumping lemma, $w' \in L$)

In case ①, $w' \notin L$, due to v?

In case ②, $w' \notin L$, due to y? also, no. of a's, b's $\neq$ no. of c's

Case ③ → v, y use only 1 symbol: (a) or (b) or (c)

↳ very similar to regular pumping lemma.

~~v y~~

~~① a...a b...b c~~

v y

① a...a a...a

② b...b b...b

③ c...c c...c

For all cases, No. of a's, b's, c's won't be conserved.

→

Date / /

Ex) $L = \{a^n \mid n \geq 1, n \text{ is prime}\}$

~~Let L~~

Assume L is a CFL, and pick some N.

choose $|w| > |\phi(a)^{|N|}|$, $(w \in L)$

So, $w = a^p \mid p$ is prime, $p > \phi(a)^{|N|}$

Assume $p = a_1 + b_1 + c_1 + d_1 + e_1$ $(b_1 + d \neq 0,\ a_1, b_1, c_1, d_1, e \geq 0)$

$u = a^{a_1}$, $v = a^{b_1}$, $w = a^{c_1}$, $y = a^{d_1}$, $z = a^{e_1}$

$w = uvxyz \in L$.

Consider ~~Let~~ $k = a_1 + c_1 + e_1$.

Let us pump v, y : 'k' times

$\Rightarrow w' = uv^k x y^k z \mid k = a_1 + c_1 + e_1$

By pumping lemma for CFLs, $w' \in L$.

However, $|w'| = a_1 + (a_1 + c_1 + e_1)b_1 + c_1 + (a_1 + c_1 + e_1)d_1 + e_1$

$= (a_1 + b_1 + e_1)(1 + b_1 + d_1)$

not prime! $(b_1 + d_1 \geq 1)$

$\Rightarrow w' \notin L$

$\Rightarrow$ Contradiction

$\Rightarrow$ L is not CFL.

Method 2:

$w = uvxyz \quad vy \neq \epsilon$

let $|vy| = q > 0 \Rightarrow |uxz| = p - q$.

$\Rightarrow w = a^{(p-q)} a^{q}$

$\Rightarrow w' = a^{p-q} a^{(p-q)q}$

$\Rightarrow w' = a^{(p-q)(q+1)}$

① $q+1 \geq 1$: Trivial, $q > 0$

② $p - q \geq 1$ (?)

↳ $p \geq q+1$ $\boxed{\geq 1}$

~~p≥2 is closing~~

$p \geq 2$

$q \geq 1$

$\Rightarrow p - q \geq 1$

But $|p - q \neq 1|$ (?) is reqd.

For $p = q+1$, ~~only~~ ~~p≥2~~

$|uxz| = 1$

↓

pump vy zero times and $|w| = 1 \neq$ prime

→ closure under intersection

Given L_1, L_2 are CFLs. Is $L_1 \cap L_2$ a CFL?

let $L_1 = \{a^n b^n c^m, n, m \geq 0\}$ → #a's = #b's
$L_2 = \{a^m b^n c^n, n, m \geq 0\}$ → #b's = #c's

$L_1 \cap L_2 \Rightarrow \{a^n b^n c^n, n \geq 0\}$ → #a = #b = #c

We know $L_1 \cap L_2$ is not CFL here

⇒ CFLs are NOT closed under intersection disproved via counterexample.

→ closure under complementation

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

If CFLs are closed under complementation,
→ CFLs are closed under intersection
⇒ $(p \rightarrow q)$ ≡ $(\neg q \rightarrow \neg p)$
We know q is false,
i.e. $\neg p + q = 1$
$\Rightarrow (p = 0)$

⇒ CFLs are NOT closed under complementation.

→ This is all an effect of the non-determinism of CFLs

→

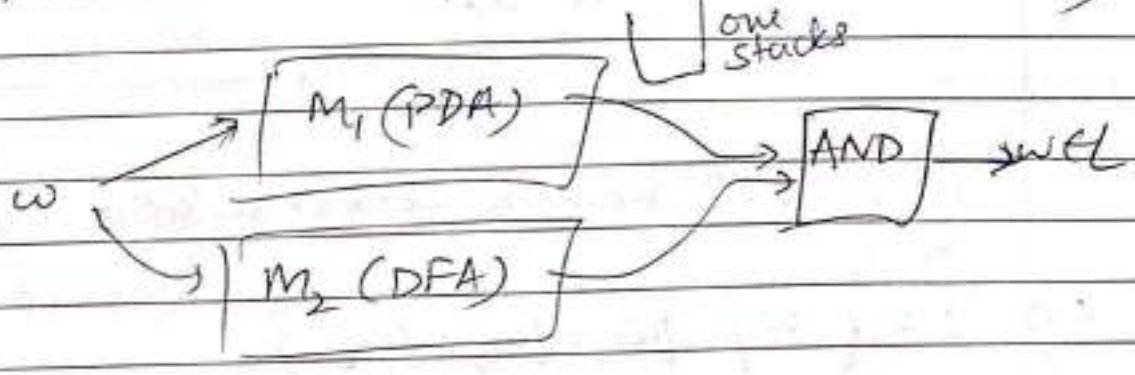
* Shuffle of

DOMS | Page No.
Date / /

Closure under intersection w/ Regulars :

CFL, L_1 : has CFG / PDA
Regular, L_2 : has DFA.
} Is $L_1 \cap L_2$, CFL?

(Product Automata of L_2, L_1 would give a setup for intersection of the machines.



Consider :-

$M_1 = (Q_1', \Sigma', \Gamma_1', \delta_1', q_0', F_1')$

$M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$

$\downarrow$

$M = (Q, \Sigma, \Gamma_\emptyset, \delta, q_0, F)$

$Q = Q_1 \times Q_2$ $\quad q_0 = (q_0', q_0^2)$

$\Sigma = \Sigma$ $\quad F = F_1 \times F_2$

$\Gamma = \Gamma$

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

(rough)

read from word ($\in \Sigma$) ; read from stack ($\in \Gamma$)

$M_1 : (q_1, a, B) \to (p_1, C) \quad \forall q_2 \in Q_2$

$M : ((q_1, q_2), a, B) \to (\delta_1(q_1, a, B), \delta_2(q_2, a))$

DOMS | Page No.
Date / /

$M_1 : (q_1, \epsilon, B) \rightarrow (p_1, c)$

$\forall q_2 \in Q_2$

$M : ((q_1, q_2), \epsilon, B) \rightarrow ((p_1, q_2), c)$

NPDA, M'

(L = $a^n b^n c^n$ would be "context-sensitive")

Ex) $L = \{ w \mid n_a = n_b = n_c \}$

If L is CFL, $L \cap L_R$ is CFL, where L_R is regular

let $L_R = a^* b^* c^*$

$L \cap L_R = a^n b^n c^n \rightarrow$ NOT CFL

Hence, L is NOT CFL

x

DOMS | Page No.
Date 18/21/22

TY

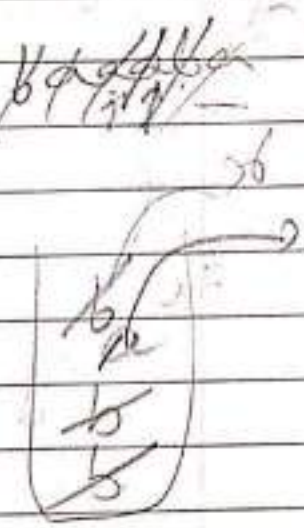
1. *

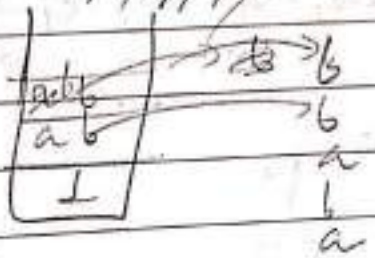
$L = \{ w \in \{a, b\}^* \mid \#|w|_a = 2|w|_b \}$

$(q, b, I) \rightarrow push(bb)$
$(q, a, I) \rightarrow push(a)$

$(q, a, a) \rightarrow push(a)$
$(q, b, b) \rightarrow push(bb)$

$(q, a, b) \rightarrow pop(b)$
$(q, b, a) \rightarrow pop(a), push(b)$





baaaab

011110

2. Show that PL is context free:

L1: $L_1 = \{a^{n^2} : n \geq 0\}$

$w = a^{n^2}$

$w = a^{a_1} a^{b_1} a^{c_1} a^{d_1} a^{e_1}$

$n^0 = a_1 + b_1 + c_1 + d_1 + e_1$

$w' = a^{(a_1 + k b_1 + c_1 + k d_1 + e_1)}$

To prove $a_1 + k b_1 + c_1 + k d_1 + e_1$ is not a square for some k.

$k = 0$: $|w'| = a_1 + c_1 + e_1$

P-1: Okay. L_1 is CFL

P-2: FINE. Here you go. $(\phi(c))^N$

P-1: Take w: $|w| > |\phi(c)|^{n+1}$

$w = a^{n^2} \mid n^2 > |\phi(c)|^{n+1}$

P-2: Pls stop ty. $w = uvxyz$

$|vy| \neq 0$

$|vy| = k$

$vy = a^k$

DOMS | Page No.
Date / /

$w = a^{n^2-k}\,a^{k}$ (Since it's only a's, $uvwxyz = uvzxy$)

$\Rightarrow (1 \le k \le n^2)$

$k=1$: $w = a^{n^2-1}\,a^{1}$

jump zero

$|w| = n^2-1 < n^2$, $(n-1)^2 < n^2-1 < n^2$

$k=n^2$: $w = a^{n^2}$, $uxz = \epsilon$

$w' = a^{2n^2}$

$|w| \neq k^2$ $\forall k \in \mathbb{N}$

$1 < k < n^2$:

$a^{n^2-k}\,a^{k}$

$(n-1)^2 \quad n^2-k \quad n^2$

$n^2-2n+1 < n^2-k < n^2-1$

$2n-1 > k \ge 1$

$1 < k < n^2$

$\Rightarrow 0 < n^2-k < n^2-1$

$a^{n^2-k}\,a^{k}$

$a^{n^2-2k+k^2}$

$a^{n^2+k^2-k}$

$a^{(n^2+k^2+2k)-k}$

$(n+k)^2-k$ $\quad 1 < k < n^2$

DOMS | Page No.
Date / /

3. Closure under union for context free.

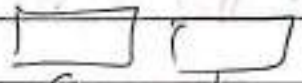
L: $L = \{ a^m b^n : m \neq n \}$

$L_1 = \{ a^n b^n : m = n \}$ → context free

$L'' = \{ a^* b^* \}$ → context free

$L_1 \cup L' \rightarrow L''$

CFL CFL



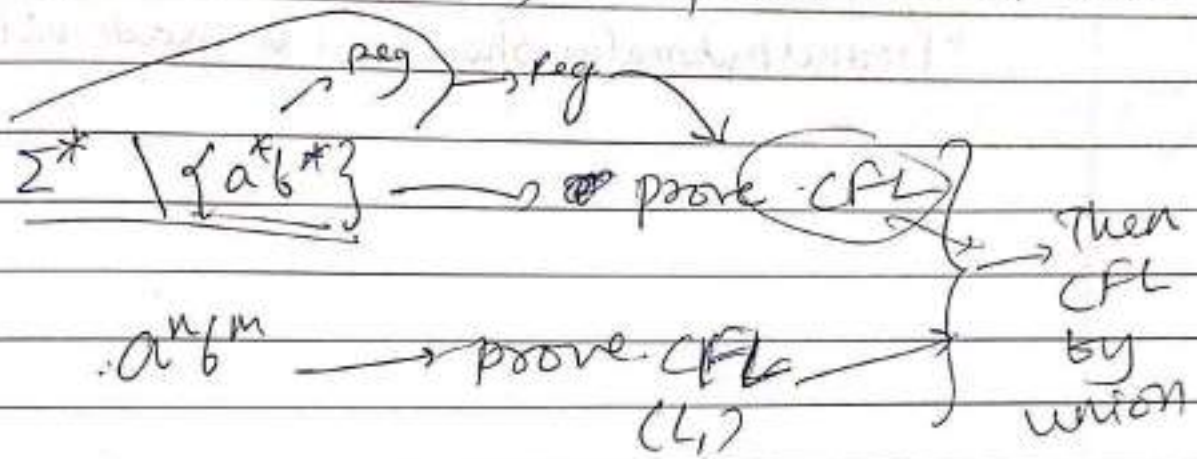
$m \neq n$ → $m > n$ $L_2 = \{ a^m b^n / m > n \}$

→ $m < n$ $L_3 = \{ a^m b^n / m > n \}$

DOMS | Page No.
Date / /

Ex: $\{a,b\}^* \setminus \{a^n b^n : n \geq 0\}$

For CFLs with DPDA, complementation works



DOMS | Page No.
Date / /

4) CFLs: closed under intersection w/ regulars.

DFAs ~~Intersection~~
↳ why not NFA?

"Product Automatas should not be made using NPAs"

DOMS | Page No.
Date / /

Ans) $L = \{ww : w \in \{a,b\}^*\}$

$w = a^n b^n a^n b^n$

$uvxyz$

~~$|vxy| \geq p$~~

$n \geq \frac{|\phi(G)|^{|N|}}{4}$

$|vxy| \leq \phi(G)^{|N|}$

Outline

- Turing Machine
- IDs : config
- Exs : $a^n b^n$, $a^n b^n c^n$

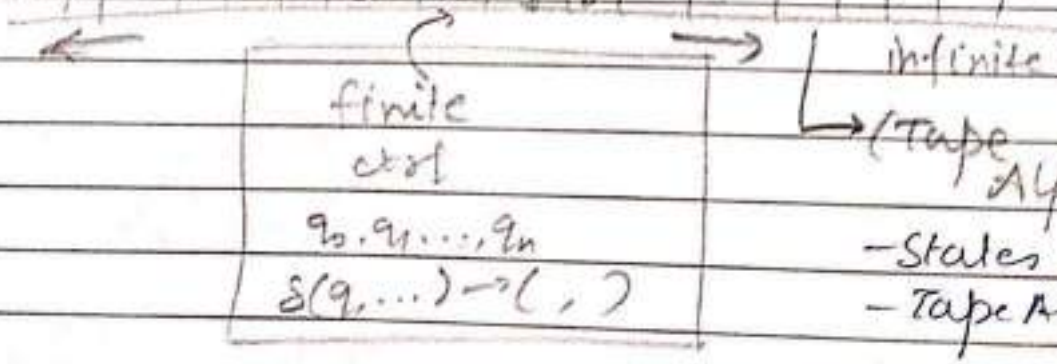
- Arith f^ns
  + +, −, ×, subroutines

→ Turing Machines

- NFAs / DFAs → RegEx → RegLang } "Restricted" TM.
- PDA - 1 stack : CFGs → CFLs.
- $a^n b^n c^n$ - 2 stacks : context sensitive langs (CSL) (of CSG's)
- General TMs
  ↳ End of course.

→ Chomsky hierarchy of formal langs.

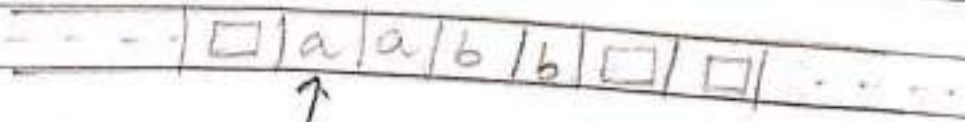
→



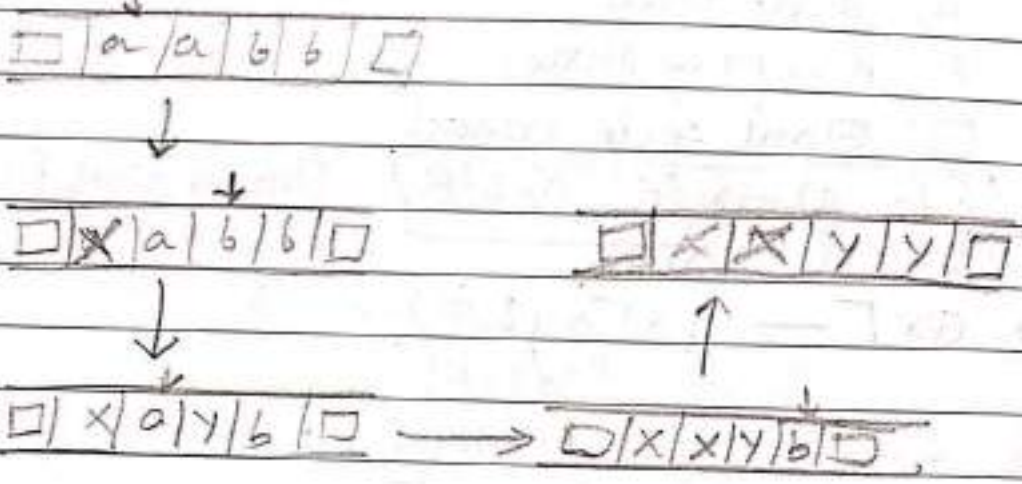
- States
- Tape Alphabet.

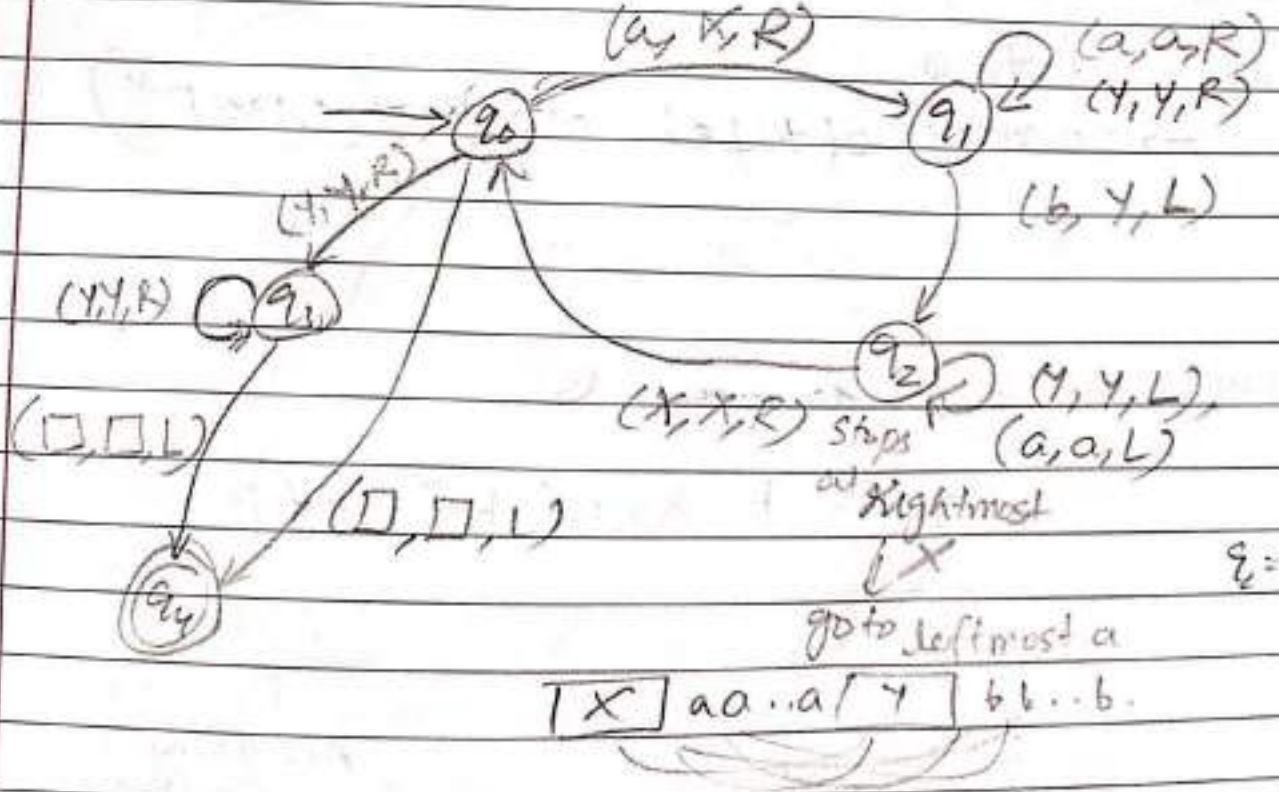
- When finite ctrl is reading a cell on the tape, it is said to be "scanning" the cell.
- The tape is R/W tape, can be edited, read, written.
- We can move from one cell to another, to left, or to right

Ex) $L = \{a^n b^n \mid n \geq 0\}$ $L \in$ CFLs, $L \notin$ REs



when reading a, replace w/ X.
when reading b, replace w/ Y.





For $a^n b^m$,

- $n > m$: stuck at q_1
- $m > n$: stuck at q_3
- $m = n$: goes to q_4

stuck at q_2: system error

→ Formally, Turing machine is,

$$M = (Q, \Sigma, \Gamma, q_0, F, \square, \delta)$$

where, Q: set of states
Σ: input alphabet
Γ: Tape alphabet $\Sigma \subseteq T$
q_0: start state
F: Accepting state
□: Blank cells symbol
$\delta(q, a) \rightarrow (q', X, L/R)$ Transition fx

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times (L/R)$$
$$\hookrightarrow \{L, R\}$$

→ Configuration of Turing Machine:
↓
"Complete Description"

● → State ● in Q
→ contents of tape: $\alpha q \beta$, $(\alpha, \beta \in \Gamma^*)$

$$\ldots \alpha | q | \beta \ldots$$

For some word

$$q_0 aw' \vdash X q_1 w' \vdash^* \alpha f \beta$$

□	q_0	a	w'	□	X	q_1	w'	α	f	β

↑ Initial config

↑

↑ Accepting config

"Stop at final state"

→ Infinite loop is possible. } If $w \notin L(M)$
~~Non~~ Termination / "stuck" due to no possible opⁿ

$(q_0 w \vdash^* \alpha q \beta)$

(i.e.) If $w \notin L(M_t)$, $q_0, q' \in Q$

$q_0 w \vdash^* \alpha q' \beta$: $q' \notin F$, $\alpha, \beta \in \Gamma^*$, $w \in \Sigma^*$

(or)

$q_0 w \vdash^* \ldots$: Infinite loop

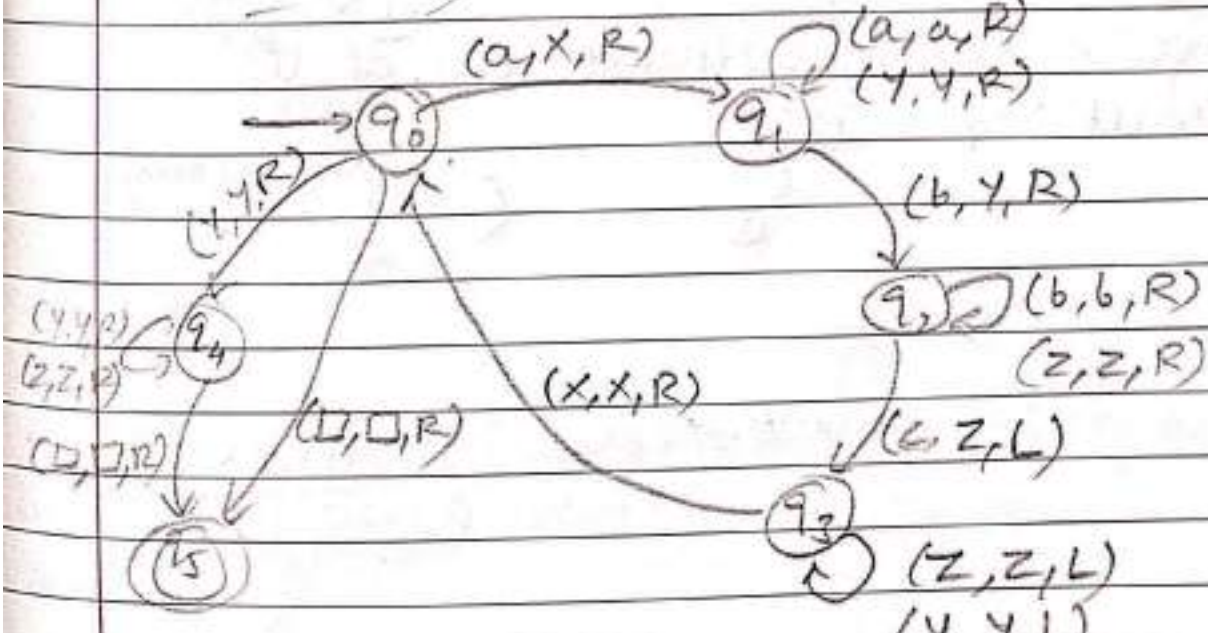
Ex) $L = \{ a^n b^n c^n \mid n \geq 0 \}$

$a^n b^n c^n$

Then, $x^n y^n c^n$

use $a^n b^n$ machine.

can generalize to $a_1^n a_2^n \ldots a_k^n$



$a = b = c \longrightarrow q_0$

$a = b < c \longrightarrow q_4$ $\quad a < b < c$

$a = b > c \longrightarrow q_2$ $\quad a < c < b$

$b = c > a \longrightarrow q_4$ $\quad b < c < a$

$b = c < a$ $\quad b < a < c$

$a = c < b$ $\quad c < a < b$

$a < c > b$ $\quad c < b < a$

DOMS | Page No.
Date / /

→ Turing Machine to add two Numbers

- Can some problem given in firstorder logic be proven? true/false
  - Entscheidungsproblem.
  (Decision problem)

$\phi \rightarrow$ [Algo] $\rightarrow$ Y / N

- Approaches:
  - lambda calculus (via church)
  - TMs. (via guess who)
  - Gödel (Proof of incompleteness)

- Numbers are given in unary, i.e., $(0^n)_1 = (n)_{10}$ for $\Sigma = \{0\}$

Ex) $0^n 1 0^m$ 1 = separator

(q0, n, m)

Initially, reading leftmost zero of 0^n.

Result reqd: 0^{n+m} (q1)

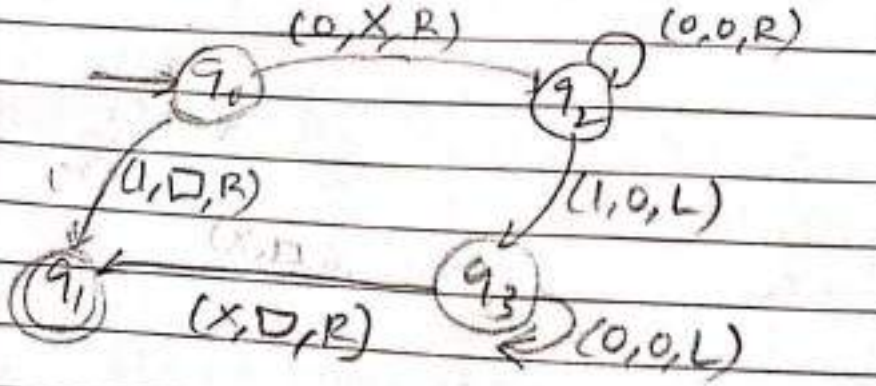
$(0^n 1 0^m) \vdash^* (0^{n+m})$

0 0 1 0 0 0

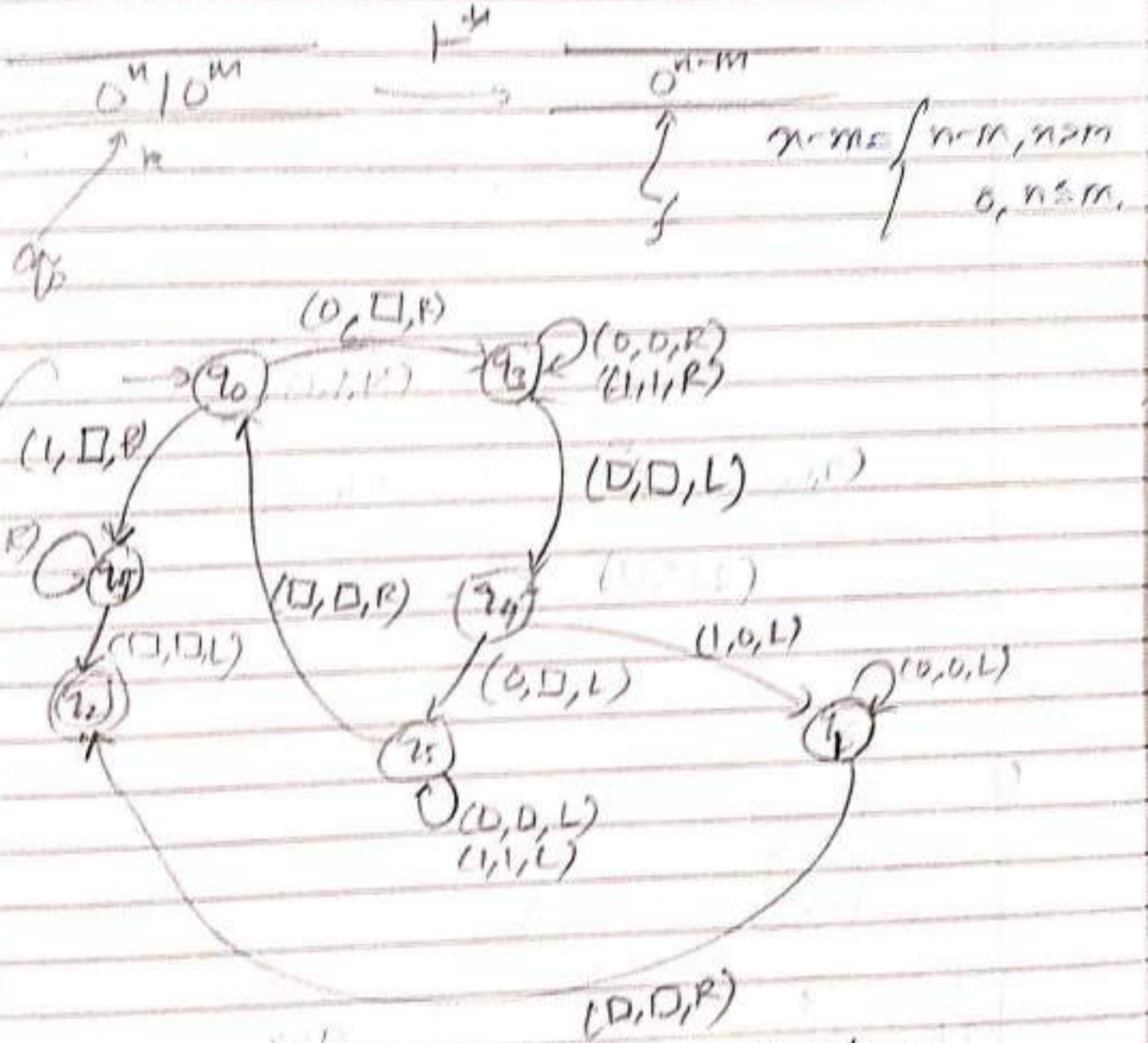
↑

go to 1, replace it w/ zero,

go back to leftmost 0, wipe it off the face of the earth



→ Turing Machine to subtract two numbers



→ Turing Machine to multiply two numbers