Parse Tree: compiler builds the parse tree

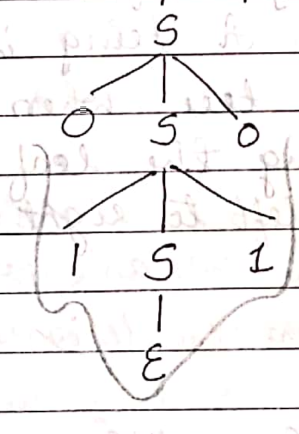
A parse tree is a tree representation of a derivation. The tree will be labelled tree with the following condition

(i) Root of the tree will always be start symbol.
(ii) All non leaf nodes of the tree will be nonterminals
(iii) All leaf nodes of the tree will be terminals.
Note: ε (Epsilon) can be leaf but it has to be the only child to its parent.

If A is a non terminal and $x_1, x_2 \ldots x_n$ are its children starting from the left then there will be a production
$A \rightarrow x_1 x_2 \ldots x_n$ in the grammar.

Q: Construct a parse tree for the string 0110 in the language of a palindrome

Grammar
$S \rightarrow 1S1 \mid 0S0 \mid 0 \mid 1 \mid \varepsilon$



$\Rightarrow 01\varepsilon10$
$\Rightarrow 0110$ — Yield of parse tree.

Root vertex: Must be labelled by the start symbol.
Vertex: Labelled by Non terminal symbols
Leaves: Labelled by terminal symbol or ε.

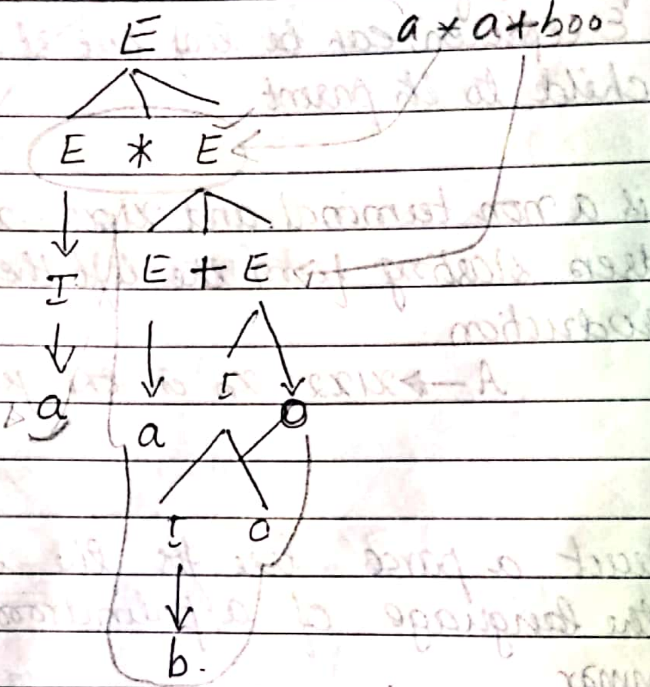
Date :
Page No :

Q: Construct the parse tree for the string $a*a+b00$ using grammar using the grammar

$E \to I$

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

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



$\therefore$ $a*a+b00$ – yield of parse tree.

Yield of parse tree: A string is said to be a yield of a parse tree when we obtain it by concatenating the leaf nodes of parse tree from left to right.

5.1.2 The grammar $S \to AS/\epsilon$, $A \to aa/ab/ba/bb$.
Give leftmost and rightmost derivations for the following:

(a) aabbba

$S \underset{lm}{\Rightarrow} AS \Rightarrow aaS \Rightarrow aaAS \Rightarrow aabbS \Rightarrow aabbAS \Rightarrow aabbbaS \Rightarrow aabbba$

$S \underset{rm}{\Rightarrow} AS \Rightarrow AAS \Rightarrow AAAS \Rightarrow AAA\epsilon \Rightarrow AAba \Rightarrow Abbba \Rightarrow aabbba$

Q Obtain the string a*b+a using the following grammar

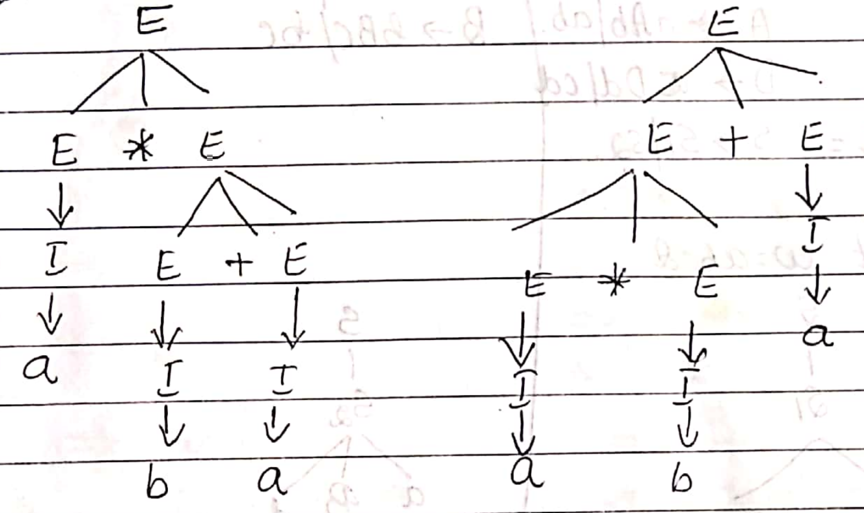
$E \to I$

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

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

Soln: Parse tree.

Lmd. Rmd.



∴ Yield = a*b+a ∴ a*b+a

Yield of both parse tree = a*b+a.

Ambiguous grammar:

For the same string if we are able to write more than one parse tree, then the grammar is said to be ambiguous grammar

A grammar G = (V,T,P,S) is said to be ambiguous if there is any string w belonging to T* for which we can write more than one parse tree with the root labelled 'S' & yield 'w'

Inherently ambiguous language: A language is said to be inherently ambiguous if we can not write any grammar for this language so that it is unambiguous.

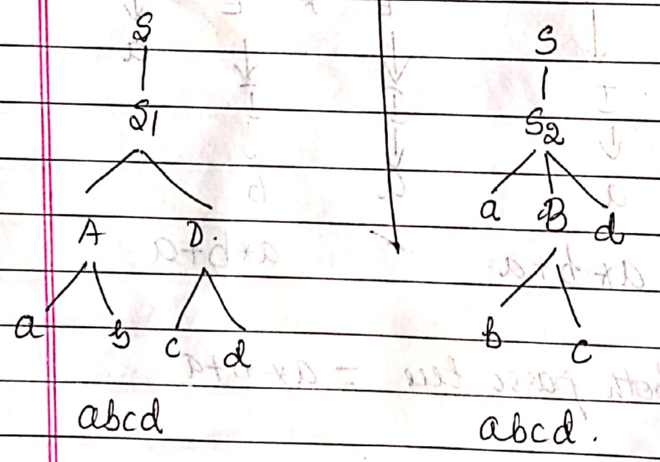
Q : Write a grammar for $L = a^m b^m c^n d^n \mid m \geq 1, n \geq 1 \cup \{a^m b^n c^n d^m \mid m \geq 1, n \geq 1\}$.

Soln : $S_1 \rightarrow AD$
$A \rightarrow aAb \mid ab$
$D \rightarrow cDd \mid cd$

$S_2 \rightarrow aS_2d \mid aBd$
$B \rightarrow bBc \mid bc$

$L = S \rightarrow S_1 \mid S_2$.

let $w = abcd$



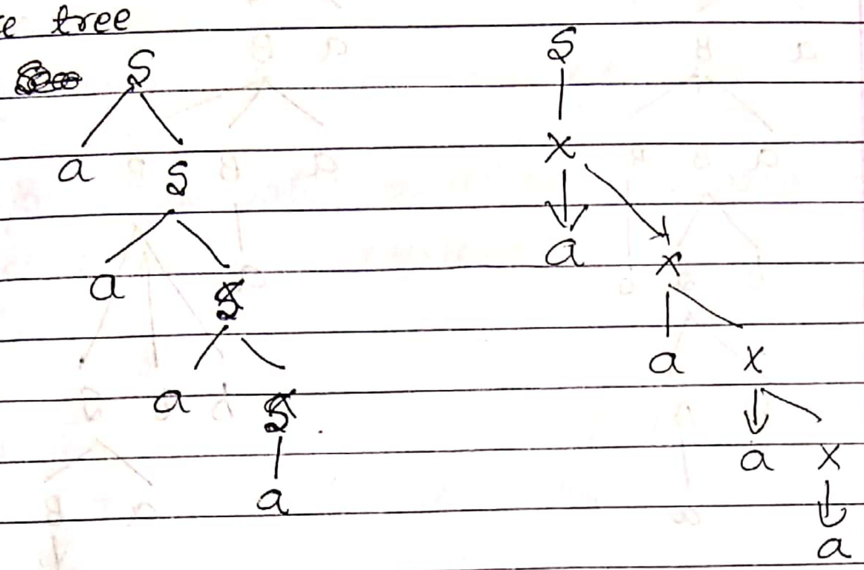
As per the above example we cannot write unambiguous grammar if it is inherently ambiguous grammar.

Q: Is the following grammar ambiguous:
$S \rightarrow aS \mid X$
$X \rightarrow SaX \mid a$

Date :
Page No :

Soln: let $w = aaaa$.

Parse tree



as $S \overset{lm}{\Rightarrow} aS$ $\qquad$ $S \Rightarrow X$

$\Rightarrow aaS$ $\qquad$ $\Rightarrow aX$

$\Rightarrow aaaS$ $\qquad$ $\Rightarrow a\,aX$

$\Rightarrow aaaa$. $\qquad$ $\Rightarrow a\,a\,aX$

$\Rightarrow a\,a\,a\,a$

aaaa

As structure of parse tree are different left most derivatives are different for a string $w = aaaa$ it is ambiguous

Q: Is the following ambiguous

$S \to aB \mid bA$

$A \to aS \mid bAA \mid a$

$B \to bS \mid aBB \mid b$

Soln: $w = aabbab$ (Any string can be taken)

$S \overset{lm}{\Rightarrow} aB$ $\qquad$ $S \overset{lm}{\Rightarrow} aB$

$\Rightarrow aaBB$ $\qquad$ $\Rightarrow aaBB$

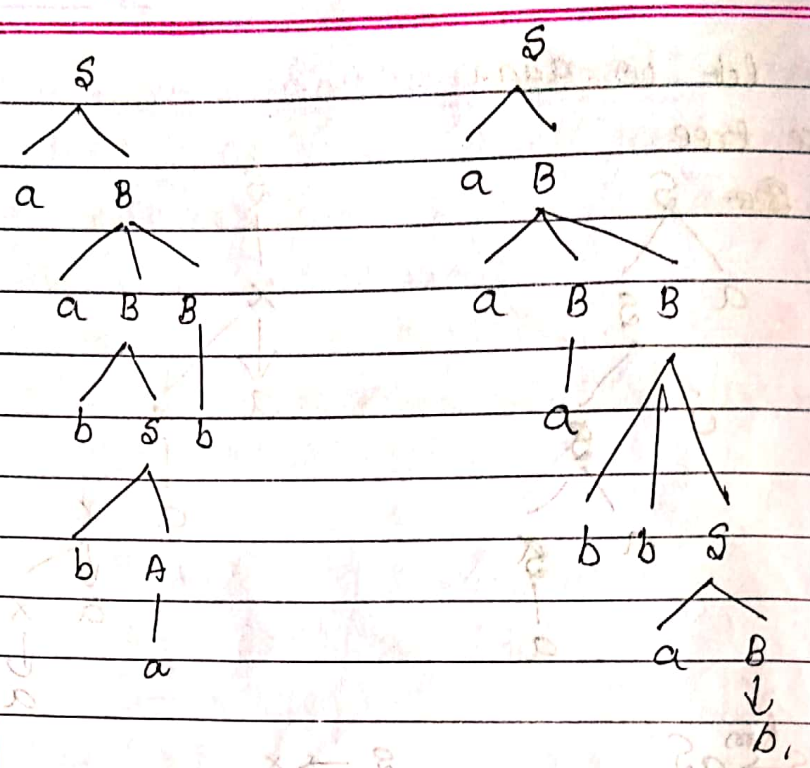
$\Rightarrow aabSB$ $\qquad$ $\Rightarrow aabB$

$\Rightarrow aabbAB$ $\qquad$ $\Rightarrow aabbS$

$\Rightarrow aabbaB$ $\qquad$ $\Rightarrow aabbaB$

$\Rightarrow aabbab$ $\qquad$ $\Rightarrow aabbab$

Scanned with CamScanner



The given grammar is ambiguous.

Q Obtain the string aaabbabbba by applying leftmost derivation and the parse tree for the grammar shown below. Is it possible to obtain the same string again by applying leftmost derivation derivation but by selecting different productions?

$S \to aB \mid bA$

$A \to aS \mid bAA \mid a$

$B \to bS \mid aBB \mid b$.

Soln:

$S \Rightarrow aB$	$S \Rightarrow aB$
$\Rightarrow aaBB$	$\Rightarrow aaBB$
$\Rightarrow aaaBBB$	$\Rightarrow aaaBBB$
$\Rightarrow aaabSBB$	$\Rightarrow aaabBB$
$\Rightarrow aaabbABB.$	$\Rightarrow aaabbSB$
$\Rightarrow aaabbaBB$	$\Rightarrow aaabbaBB$
$\Rightarrow aaabbabSB$	$\Rightarrow aaabbabB$
	$\Rightarrow aaabbabbS$

$\Rightarrow aaabbab\ b\ B$ $\qquad\qquad \Rightarrow aaabbabbbA$

$\Rightarrow aaabbabb\ bB$

$\Rightarrow aaabbabb\ b$ $\qquad\qquad \Rightarrow aaabbabbba$

As leftmost derivates are distinct the parse tree constructed will be different.

Q: Is this grammar ambiguous

$S \rightarrow AB / aaB$

$A \rightarrow a / Aa$

$B \rightarrow b$

Soln: Let $w = aab$.

$S \rightarrow AB$

$\Rightarrow aaB$

$\Rightarrow aab$

$S \Rightarrow aaB$

$\Rightarrow aab$

The grammar is ambiguous.

Q: Show that the following grammar is ambiguous

$S \rightarrow aSbS / bSaS / \epsilon$

Soln: Let $w = abaabb$.

$S \Rightarrow aSbS$

$\Rightarrow abSaSbS$

$\Rightarrow aba\epsilon SbS$

$= abaSbS$

$\Rightarrow aba\,aSbSbS$

$= abaa\epsilon bSbS$

$abaab\epsilon bS$

$abaabbS$

$abaabb\epsilon$

$abaabb$

$S \Rightarrow aSbS$

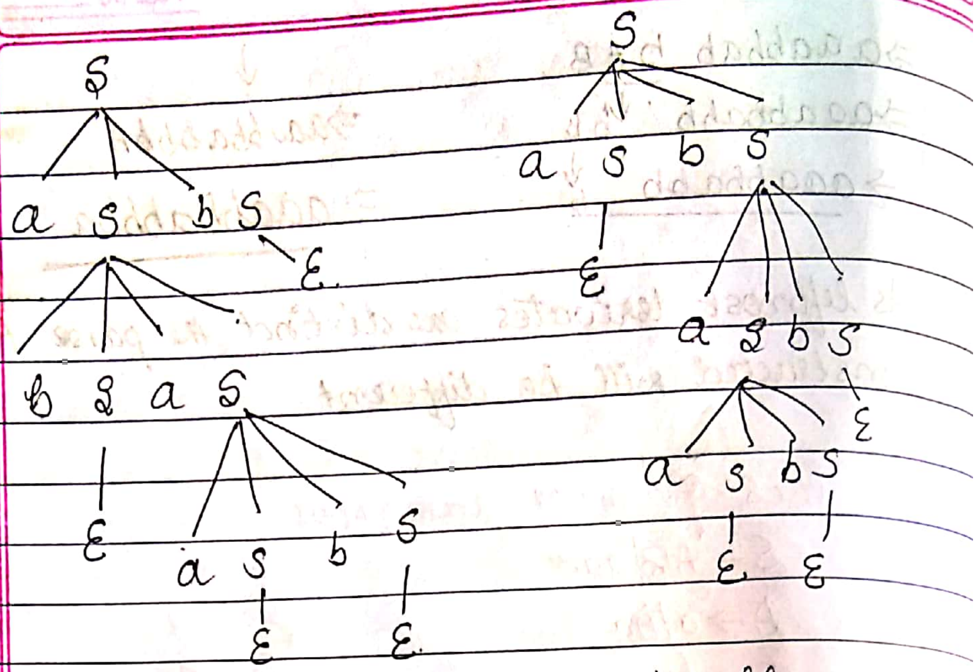
$S \Rightarrow a\epsilon bS$

$S \Rightarrow abS$

$S \Rightarrow ab\,aSbS$

$S \Rightarrow aba\,aSbSbS$

$S \Rightarrow abaabb$

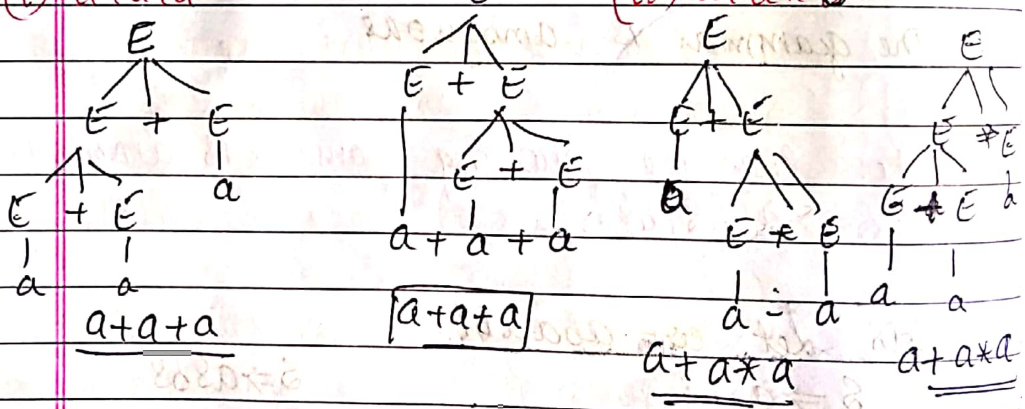


abaabb,

~~aab~~
abaabb
Grammar is ambiguous

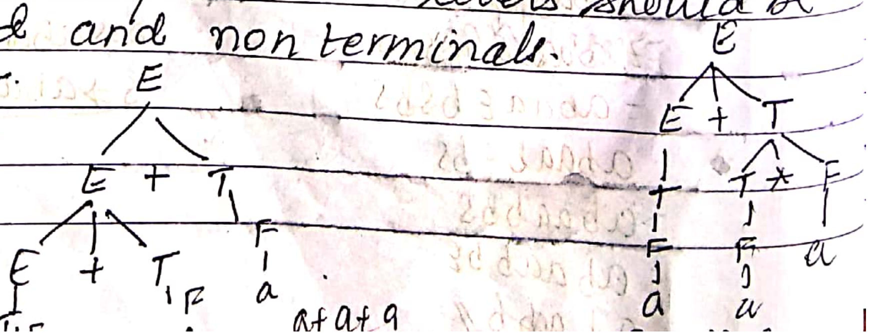
Ambiguous:
$E \to E+E$
$E \to E*E$
$E \to a.$

(i) a+a+a   (ii) a+a*a



Unambiguous: To remove ambiguity follow associativity using left recursion.
To have precedence levels should be introduced and non terminals.

$E \to E+T/T$
$T \to T*F/F$
$F \to a.$



Removing ambiguity in grammars.

* The reason for ambiguity in grammar is because the precedence of operator are not taken into consideration. If we can force the precedence then we can make grammar unambiguous.

* To obtain an unambiguous grammar we can add additional nonterminals to force the precedence

1. Factor is an expression that can not be broken apart by any adjacent operator, either a* or a+.

The only factor in our language are

(i) Identifier: It is not possible to separate the letters of an identifier by attaching an operator (All terminals)
(ii) Any paranthesis expression no matter what appears inside prevents from becoming the operand of any operator outside the parenthesis

2. A term is an expression that cannot be broken by + or - operator. It has expression of highest precedence that make use of terms & factors.
Term is a product of one or more factors.

3. Expression: They are either be terms or recursively defined using expressions and terms for lower precedence operators

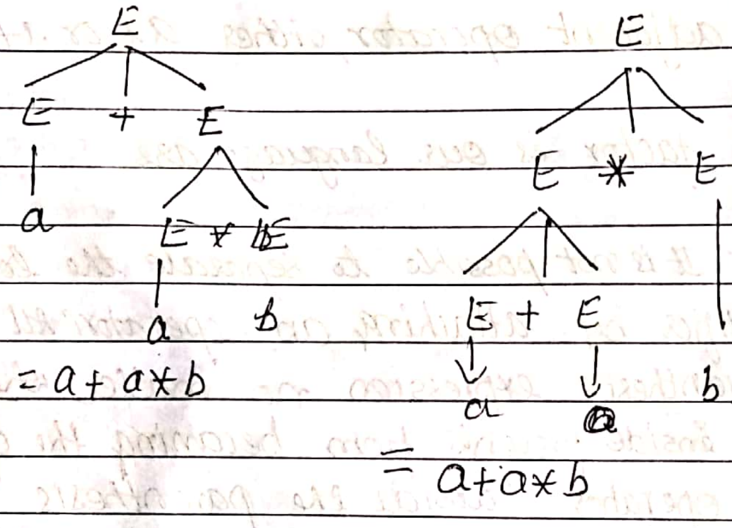
* takes precedence over +.
associativity problems removed by using left recursion
precedence is solved (by using level
+ & * are left associative.

→ left recursion

Q: Convert the following grammar to unambiguous grammar if the grammar is ambiguous.

$E \to E+E$
$E \to E*E$
$E \to E/E$
$E \to (E)/a/b$

for in $a+a*b$.



So above grammar is ambiguous.
So let us remove the ambiguity

Step 1: Group all terminals under identifier (a, b – name it I)
Step 2: Write factors
Step 3: Highest precedence operator (T, F) *, /
Step 4: Lower precedence. (using T & E) +

$I \to a/b$.
$F \to I | (E)$
$T \to F | T*F | T/F$
$E \to T | E+T$

$E \to E+T/T$
$T \to T*F/T/F/F$
$F \to I / (E)$
$I \to a/b$.

Date :
Page No :

Q: Show that the following grammar is ambiguous & write an unambiguous grammar

$S \rightarrow S+S \mid S*S \mid S-S \mid b \mid c \mid (S)$

Soln: $S \rightarrow T \mid S+T \mid S-T$
$T \rightarrow F \mid \cancel{F} * F$
$F \rightarrow I \mid (S)$
$I \rightarrow b \mid c$

higher precedence
4 levels

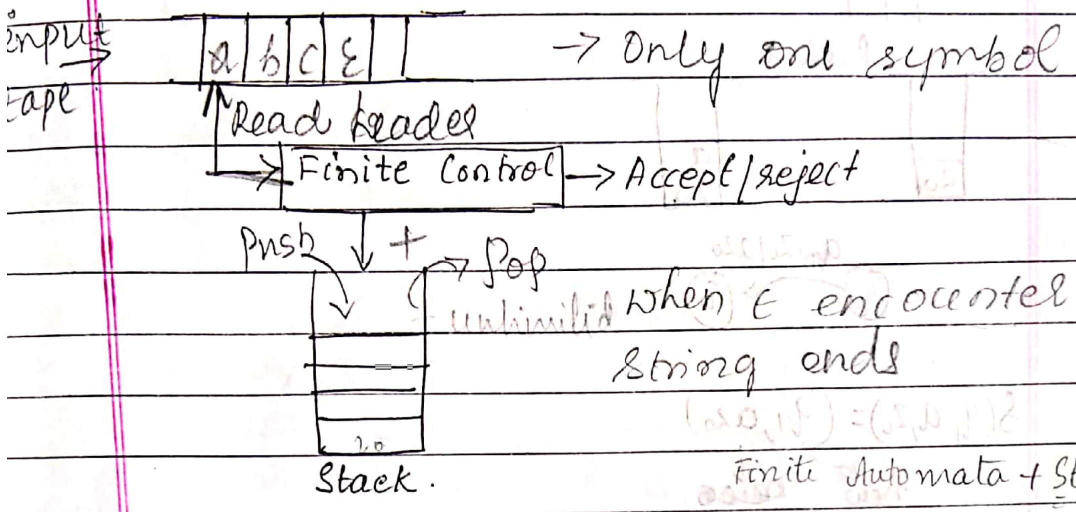
Precedence Constraints (Rule 1).
→ The level at which the production is present defines the priority of the operator contained in it
→ The higher level → lower priority
→ The lower level → higher "
→

Associativity
→ If the operator is left associative, induce left recursion.

Scanned with CamScanner

PUSH DOWN AUTOMATA (PDA).

PDA is ε-NFA with an additional stack.
It accepts context free language.



Finite Automata + Stack

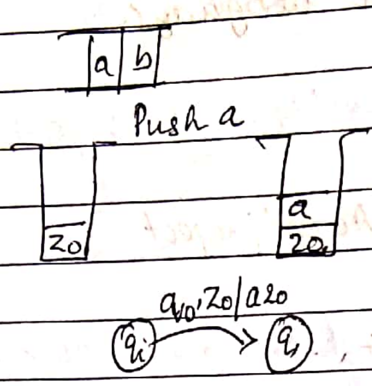
Formal definition : A PDA is defined by the 7 tuples $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$.

where Q = Finite set of states
Σ = Set of symbols that may be present in PDA (alphabet).
Γ = set of all symbols that may be present in stack. $\Sigma \subset \Gamma$ $z_0 \in \Gamma$
q_0 = Start state
z_0 = Initial symbol on stack.
F = Finite set of accepting state $F \subseteq Q$.
δ = Transition function.

It is defined as $\delta(q, a, x) = (p, r)$ — i/p symbol; ct state; state; ct top of stack

q — current state
a = input symbol.
x = current top of the stack.
P = state to which the m/c goes after processing symbol a.
$r \to ax$ if 'a' is pushed into stack
$\Rightarrow \epsilon$ if stack is popped.
$- x$ if stack is left alone.
z_0 — ~~top~~ bottom/top of stack.

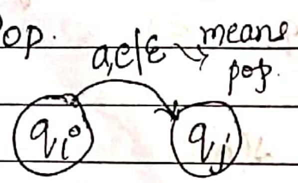
Push

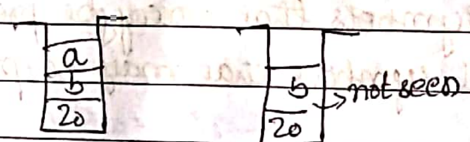


$\delta(q_1, a, Z_0) = (q_1, aZ_0)$

new state

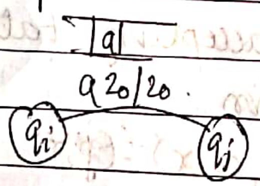
Pop

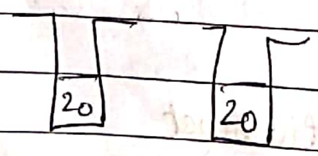




$\delta(q_i, a, c) = (q_j, \varepsilon)$

Skip





Date :
Page No :

Q1 Design a PDA for the language $L = \{a^n b^n \mid n \geq 1\}$

Soln:

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

Acceptance by empty stack

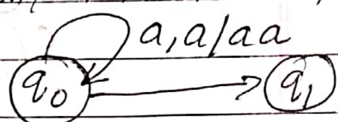
aaabbb

Since we need to identify same no. of 'a' & 'b's we keep pushing 'a's (All) then whenever 'b' is encountered pop a, if after popping all b its end of stack z_0 is accepted.

a
a
a
a
z_0

a, z_0/az_0

Step1: Push a top of stack



first line: $\delta(q_0, a, z_0) = (q_0, az_0)$

current state, i/p, next state, top of stack

2nd move $\delta(q_0, a, a) = (q_0, aa)$

$\cancel{\delta(q_0, a/a) = (q_0, aa)}$

till we encounter a we are at q_0 once b encountered → goes to q_1

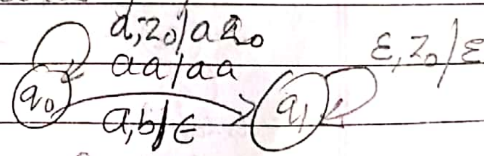
indicates pop

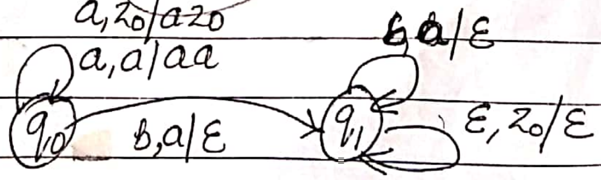
a
a
a
z_0

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

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

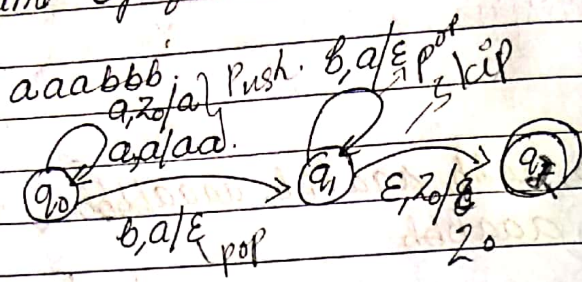
$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$] pop





Acceptance by final state.

aaabbb

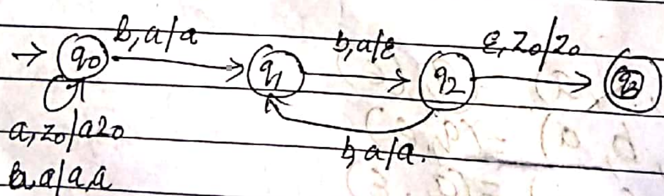


$$\delta(q_0, a, z_0) = (q_0, a z_0)$$
$$\delta(q_0, a, a) = (q_0, aa).$$
$$\delta(q_0, b, a) = (q_1, \varepsilon)$$
$$\delta(q_1, b, a) = (q_1, \varepsilon).$$
$$\delta(q_1, \varepsilon, z_0) = (q_2, z_0).$$

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, z_0\}, \delta, q_0, z_0, \{q_2\})$.

$a^n b^{2n}$.



Q2 Design a pus

accept a strin

$L = \{ abb, aa$

Soln For la secn f

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

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

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

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

$\delta(q, \varepsilon, z_0) = (q$

Transition dia

$M = \{\{q$

Q:3 Design a PD

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

$L = \{abcba,$

Soln: $\delta(q_0, a, z_0) =$

$\delta(q_0, b, z_0) =$

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

$\delta(q_0, b, a)$

$\delta(q_0, a, b)$

$\delta(q_0, b, b)$

$\delta(q_0, c, z$

$\delta(q_0, c, a$

$\delta(q_0, c,$

$\delta(q_1, a, a$

$\delta(q_1, b,$

$\delta(q_1, \varepsilon$

Scanned with CamScanner

Q2 Design a pushdown automata (PDA) to accept a string of the form $L=\{a^n b^{2n} \mid n \geq 1\}$.

$L = \{ abb, aabbbb, aaabbbbbb, \dots \}$.

Soln For 1a secon push two a's.

$\delta(q_0, a, z_0) = (q_0\ aaz_0)$ - push (skip)

$\delta(q_0\ a, a) = (q_0, aaa) \rightarrow$ push

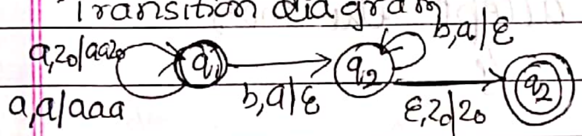
push two element

$\delta(q_0, b, a) = (q_1, \varepsilon)$ - pop

$\delta(q_1, b, a) = (q_1, \varepsilon)$ - pop

$\delta(q_1, \varepsilon, z_0) = (q_2, z_0)$ - skip

Transition diagram



$M = \{\{q_0, q_1, q_2\}, \{a, b\}, \Gamma, \delta, q_0, z_0\ \{q_2\}\}$

Q:3 Design a PDA for the language $L=\{wcw^R$ over $\Sigma=\{a,b\}^*\}$

$L = \{ abcba, aacaa, abacaba, bacab \dots \}$

Soln:

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

$\delta(q_0, b, z_0) = (q_0, bz_0)$

$w = ab$ Push

$w^R = ba$ Reverse

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

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

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

$\delta(q_0, b, b) = (q_0, bb)$

a,b push

$\delta(q_0, c, z_0) = (q_1, z_0)$

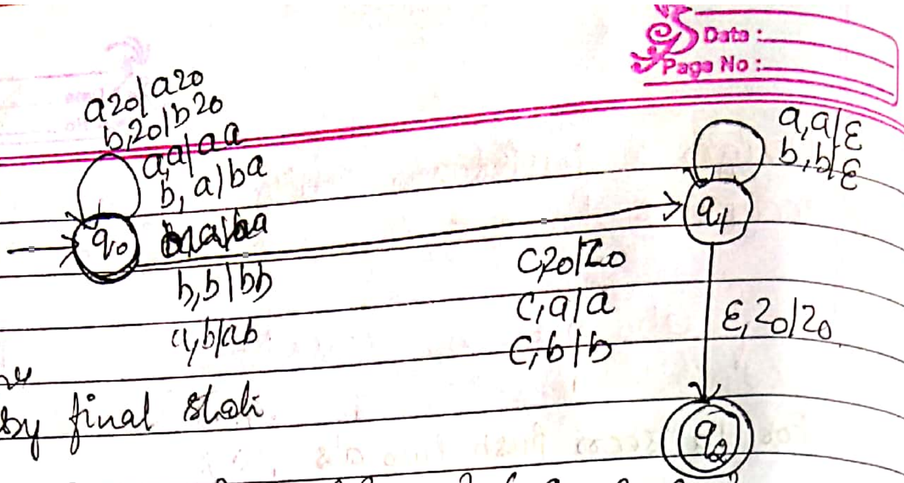
$\delta(q_0, c, a) = (q_1, a)$ middle skip

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

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

$\delta(q_1, b, b) = (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, z_0) = (q_2, z_0)$



acceptance by final state

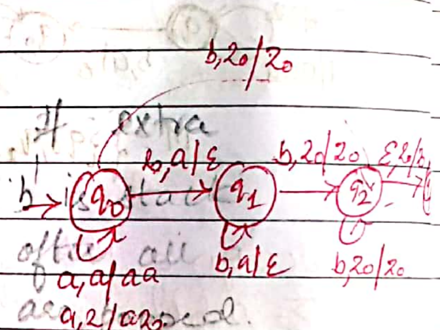
$M = \{\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, z_0\}, \delta, q_0, z_0, q_2\}$
δ as defined

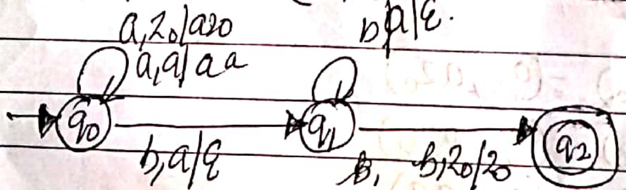
Q4 Design a PDA for the language $a^i b^j$ such that $i<j$ and $i, j \geq 1$.

Soln: Push on a's, Pop on b's & if 'b' is i/p & stack is empty accept the string

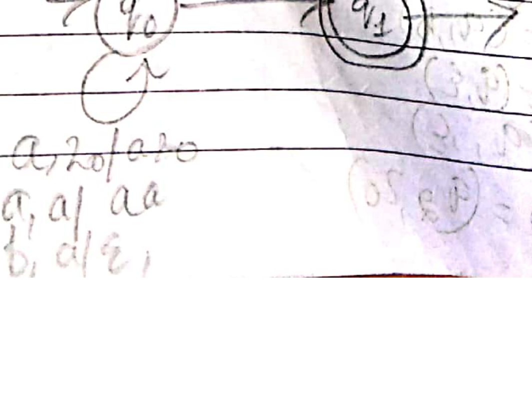
$L = \{abb, abbb \ldots\}$

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





$M = \{\{q_0, q_1, q_2\}, \{a, b\}, \{a, z_0\}, \delta$ as defined $q_0, z_0, q_2\}$,



Q5 Design a P...
L(M) = { ...

Soln:

Same sym...
Different sy...

$\delta(q_0, a, z_0)$ ...
$\delta(q_0, b, z_0)$ ...
$\delta(q_0, a, a)$ ...
$\delta(q_0, b, b)$ ...
$\delta(q_0, a, b)$ ...
$\delta(q_0, b, a)$ ...
$\delta(q_0, \varepsilon, z_0)$ ...

$M = \{\{q_0 q_1\}, \ldots$

Q.6. Design a PDA ...

Soln: $L = \{aba, abba \ldots$
This is palin...

Q5 Design a PDA that accept language
$L(M) = \{ \omega \mid \omega \in (a+b)^* \text{ and } n_a(\omega) = n_b(\omega) \}$

dpda

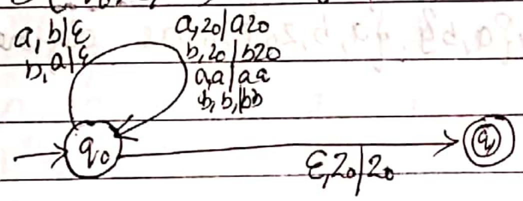
Soln:

Same symbol push (a,a)
Different symbol POP (a,b)

$$\begin{aligned}
&\delta(q_0, a, z_0) = (q_0, az_0) \\
&\delta(q_0, b, z_0) = (q_0, bz_0) \\
&\delta(q_0, a, a) = (q_0, aa) \\
&\delta(q_0, b, b) = q_0, bb)
\end{aligned} \Bigg\} \text{push}$$

$$\begin{aligned}
&\delta(q_0, a, b) = (q_0, \varepsilon) \\
&\delta(q_0, b, a) = (q_0, \varepsilon)
\end{aligned} \Big\} \text{POP}$$

$\delta(q_0, \varepsilon, z_0) = (q_1, z_0)$



$M = \{ \{q_0 q_1\}, \{a,b\}, \{a,b,z_0\}, \delta \text{ as defined}, q_0, z_0, q_1 \}.$

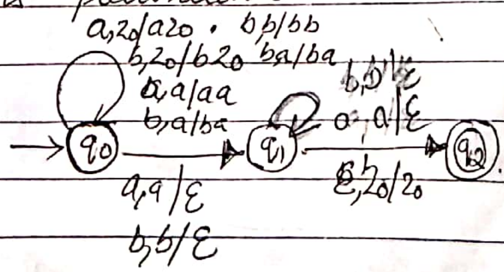
→ npda

Q6. Design a PDA that accepts language $L = \{ \omega c \omega^R \mid \omega \in (a+b)^* \}$.

even palindrome

Soln: $L = \{ aba, abba, baab, babab, \ldots \}$

This is palindrome.



Center: if it has to be center for it
Start and symbol
Should be same.
(but not definitely
abba
aaa

dead configuration &
stuck

$\delta(q_0,a,z_0) = (q_0,az_0)$ $\qquad$ $\delta(q_1,b,b) = (q_1,\varepsilon)$

$\delta(q_0,b,z_0) = (q_0,bz_0)$ $\qquad$ $\delta(q_1,a,a) = (q_1,\varepsilon)$

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

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

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

$\delta(q_0,b,b) = (q_0,bb)$

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

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

$\delta(q_1,\varepsilon,z_0) = (q_2,z_0)$

$M = \{\{q_0,q_1,q_2\},\{a,b\},\{a,b,z_0\},\delta \text{ as defined}, q_0, z_0, \{q_2\}\}$

aaaa

$q_0 \xrightarrow{a}$

$(q_0, aaaa, z_0) \rightarrow (q_0, aaa, az_0)$

centre not come $\qquad$ centre

(q_0, aa, aaz_0) $\qquad$ (q_1, aa, z_0)

$(q_0, a, aaaz_0)$ $\qquad$ (q_1, a, az_0)

$(q_0, \varepsilon, aaaaz_0)$ $\quad$ $(q_1, \varepsilon, aaz_0)$ $\qquad$ (q_1, ε, z_0) $\qquad$ $q_1\varepsilon$

q_2

Instantaneous Description.

During execution the m/c goes through a series of a series of configurations called instantaneous description given by $(q, aw, b\alpha)$

q – current state

aw – input string yet to be processed with left most symbol (a) & current i/p.

$b\alpha$ – string on the stack, with left most symbol 'b' as top of stack.

Move: It is change in instantaneous description represented by $\vdash$ (Sleeping T).

Language of PDA: A PDA accepts a string in two ways

(i) By accepting state

(ii) By empty stack.

(i) Language by accepting state: The language of PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ is given by

$L(M) = \{ w \mid (q_0, w, z_0) \vdash^*_p (q_f, \epsilon, \alpha)$ where $q_f \in F$ and stack may or may not be empty

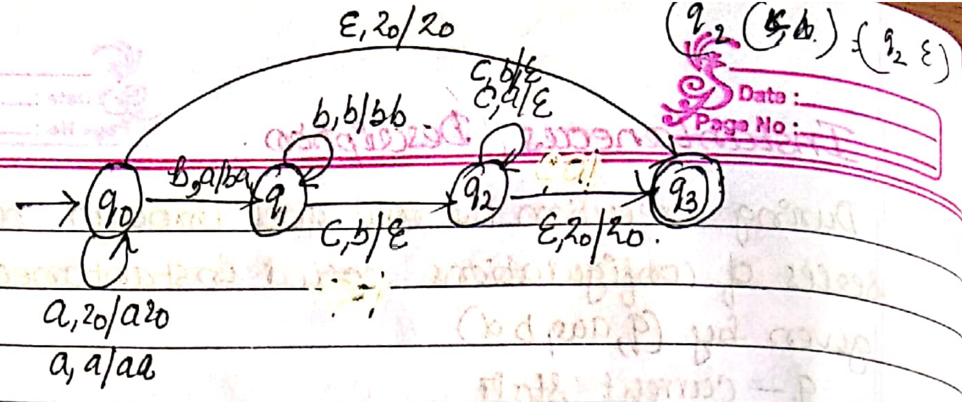
(ii) Language of PDA by empty stack:

Language of PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, \phi)$ is given by $L(M) = \{ w \mid (q_0, w, z_0) \vdash^*_p (q, \epsilon, \epsilon)$ where $q \in Q$ and stack should be empty.

Q7. Design a PDA for the language $a^n b^m c^{n+m} \mid m, n \geq 0$. Show the moves on the string abc^2.

Soln: $L = \{ abc^2, abbc^3, aabbc^4 \ldots \}$

As long as a & b are inputs push the elements when 'c' is the input pop.



$\delta(q_0 \varepsilon, z_0) = (q_0, z_0)$
$\delta(q_0 a, z_0) = (q_0, az_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0 b, a) = (q_1, ba)$
$\delta(q_1 b, b) = (q_1, bb)$
$\delta(q_1, c, b) = (q_2, \varepsilon)$
$\delta(q_2, c, a) = (q_2, \varepsilon)$
$\delta(q_2, \varepsilon, z_0) = (q_3, z_0)$

as long as input is a or b push into stack

upon 'c' pop

Moves for abc^2

$(q_0, abcc, z_0) \vdash (q_0\, bcc, az_0)$
$\vdash (q_1, cc, baz_0)$
$\vdash (q_2, c, az_0)$
$\vdash (q_2, \varepsilon, z_0)$
$\vdash (q_3, \varepsilon, z_0)$

when input ended, the m/c was in an accepting state q_3 ∴ the string abc^2 gets accepted.

Q8 Design a PDA for language $a^i b^j c^k$ $j = i+k$ $i, j \geq 0$ and moves for aabbbbcc

Soln $L = \{\varepsilon, aabbbbc, \ldots \}$

$\delta(q_0 \varepsilon, z_0) = (q_0 z_0)$
$\delta(q_0 a, z_0) \vdash (q_0, az_0)$
$\delta(q_0, b, z_0) = (q_0 bz_0)$

$\delta(q_0, a, a) = (q_0, aa)$ Push all 'a's

$\delta(q_0, b, a) = (q_1, \varepsilon)$ if b encountered

$\delta(q_1, b, a) = (q_1, \varepsilon)$ pop & go to q_1

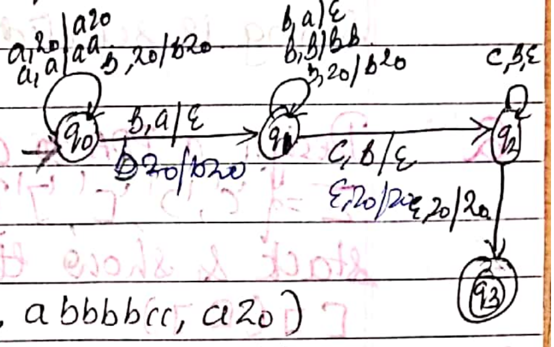
$\delta(q_1, b, z_0) = (q_1, bz_0)$ At q_1 if only b's push

$\delta(q_1, b, b) = (q_1, bb)$ if c is seen pop.

$\delta(q_1, c, b) = (q_2, \varepsilon)$

$\delta(q_2, c, b) = (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, z_0) = (q_3, z_0)$.



Moves:

$\delta(q_0, aabbbbcc, z_0) \vdash (q_0, abbbbcc, az_0)$

$\vdash (q_0, bbbbcc, aaz_0)$

$\vdash (q_1, bbbcc, az_0)$

$\vdash (q_1, bbcc, z_0)$.

$\vdash (q_1, bcc, bz_0)$

$\vdash (q_1, cc, bbz_0)$

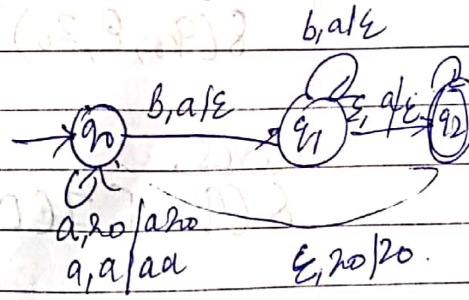
$\vdash (q_2, c, bz_0)$

$\vdash (q_2, \varepsilon, z_0)$

$\vdash (q_3, z_0)$

Q.9 Design a PDA to accept strings such that is $L = \{a^i b^j \mid i > j \ \ i, j \geq 0\}$ & show the moves on the string $a^2 b^3$.

Soln: $L = \{a, aabbb, aab \ldots\}$



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

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

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

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

$\delta(q_0, \varepsilon, a) = (q_2, \varepsilon)$ for $j = 0$

Date :
Page No :

Moves

$\delta(q_0, aabbb, Z_0) \vdash (q_0, abbb, aZ_0)$
$\vdash (q_0, bbb, aaZ_0)$
$\vdash (q_1, bb, aZ_0)$
$\vdash (q_1, b, Z_0)$

The transition on (q_1, b, Z_0) does not exist. ∴ the string is rejected

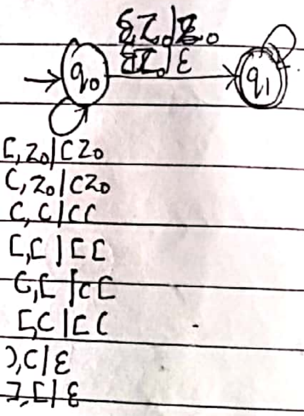
Q10 Design a PDA to accept balanced parenthesis over $\Sigma = \{ '(', ')', '[', ']' \}$. Accept the strings by empty stack & show the moves on the string [ ] ([ ])

Soln: $L = \{ [\,], (\,), [(\,)], ([\,]) \ldots \}$

USA UK

( ) – parenthesis / round
[ ] square bracket / box bracket
{ } curly bracket
< > angle brackets / braces

$\delta(q_0, [, Z_0) = (q_0, [Z_0)$
$\delta(q_0, (, Z_0) = (q_0, (Z_0)$
$\delta(q_0, (, () = (q_0, (()$
$\delta(q_0, [, [) = (q_0, [[)$
$\delta(q_0, '(', '[') = (q_0, ([)$
$\delta(q_0, '[', () = (q_0, [()$
$\delta(q_0,), () = (q_0, \epsilon)$
$\delta(q_0,], [) = (q_0, \epsilon)$
$\delta(q_0, \epsilon, Z_0) = (q_1, Z_0)$ (for accepting state
$\delta(q_0, \epsilon, Z_0) = (q_0, \epsilon)$ or (q_1, ϵ) for empty stack



Moves.

$\delta(q_0, [\,]([\,]), Z_0) \vdash (q_0,]([\,]), [Z_0)$
$\vdash (q_0, ([\,]), Z_0)$
$\vdash (q_0, [\,]), (Z_0)$
$\vdash (q_0,]), [(Z_0)$
$\vdash (q_0,), (Z_0)$
$\vdash (q_0, \epsilon, Z_0)$
$\vdash (q_0, \epsilon)$

The string is accepted becoz the stack is empty & for reaching accepting state

Q11 Design ...
a string ...
Soln: $L = \{ \epsilon, aa \ldots$

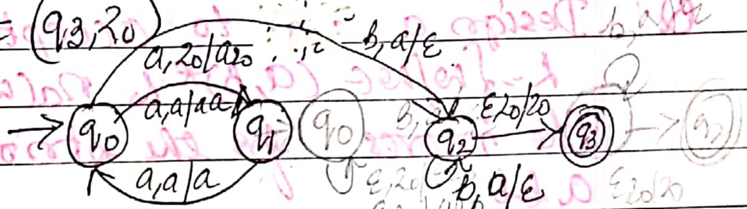
Q12 Design ...
the mov...
Soln:

Q11 Design a PDA to accept a string of them $\{a^{2n}b^n \mid n \geq 0\}$

$\delta(q_0, \epsilon, z_0) = (q, \epsilon z_0)$
$\delta(q_0, a, z_0) = (q, a z_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q, \epsilon)$
$\delta(q_1, b, a) = (q, \epsilon)$
$\delta(q_1, \epsilon, z_0) = (q, \epsilon, \epsilon)$

Soln: $L = \{\epsilon, aab, aaaabb, \ldots\}$

$\delta(q_0, \epsilon, z_0) = (q_3, z_0)$
$\delta(q_0, a, z_0) = (q_1, a z_0)$ — first a
$\delta(q_1, a, a) = (q_0, a)$ — skip second a
$\delta(q_0, a, a) = (q_1, aa)$
$\delta(q_0, b, a) = (q_2, \epsilon)$
$\delta(q_2, b, a) = (q_2, \epsilon)$
$\delta(q_2, \epsilon, z_0) = (q_3, z_0)$

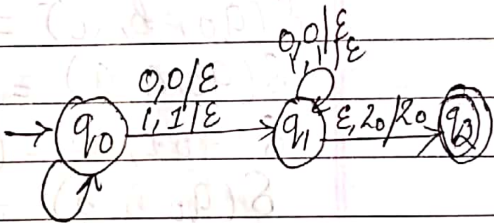


Q12 Design a PDA for $L = WW^R$ over $\Sigma = (0, 1)$. Show the moves on the string 1111.

0011

Soln:

$\delta(q_0, \epsilon, z_0) = (q_0, z_0)$
$\delta(q_0, 0, z_0) = (q_0, 0z_0)$
$\delta(q_0, 1, z_0) = (q_0, 1z_0)$
$\delta(q_0, 0, 0) = (q_0, 00)$
$\delta(q_0, 1, 1) = (q_0, 11)$
$\delta(q_0, 0, 1) = (q_0, 01)$
$\delta(q_0, 1, 0) = (q_0, 10)$
$\delta(q_0, 0, 0) = (q_1, \epsilon)$
$\delta(q_0, 1, 1) = q_1, \epsilon)$
$\delta(q_1, 1, 1) = (q_1, \epsilon)$
$\delta(q_1, 0, 0) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, z_0) = (q_2, z_0)$



Moves

$\delta(q_0, 1111, z_0) \vdash (q_0, 111, 1z_0)$

$\vdash (q_0, 11, 11z_0)$

$\vdash (q_0, 1, 1z_0)$

$\vdash (q_1, \varepsilon, z_0)$

$\vdash (q_2, z_0)$.

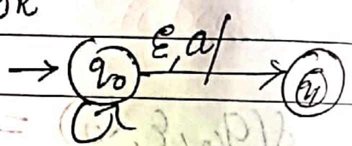
Q.13 Design a PDA to accept language $L = \{w / w \in (a, b)^* \ \& \ n_a(w) > n_b(w)\}$. Show the moves for the strings aaabb, aabba abb.

Soln: $L = \{aab, aabba, \ldots\}$

npda
$n_a(w) > n_b(w)$

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

$\delta(q_0, b, z_0) = (q_0, bz_0)$



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

$\delta(q_0, b, a) = (q_0, \varepsilon)$ a,z_0/az_0

$\delta(q_0, b, b) = (q_0, bb)$ a,a/aa

$\delta(q_0, a, b) = (q_0, \varepsilon)$, $\delta(q_0, b, a / \varepsilon) = q$

$\delta(q_1, a, z_0) = (q_2, z_0)$ b,z_0/bz_0

$\delta(q_1, \varepsilon, a) = (q_2, a)$ b,a/ε

b,b/bb

Moves

(1) aaabb

$\delta(q_0, aaabb, z_0) \vdash (q_0, aabb, az_0)$

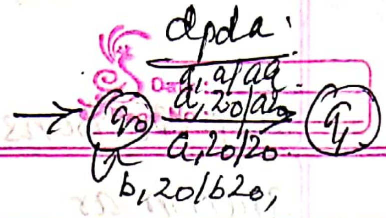
$\vdash (q_0, abb, aaz_0)$

$\vdash (q_0, bb, aaaz_0)$

$\vdash (q_1, b, aaz_0)$

$\vdash (q_1, \varepsilon, az_0)$

$\vdash (q_2, a)$



(ii) aabba

$\delta(q_0, aabba, Z_0) \vdash (q_0, abba, aZ_0)$
$\vdash (q_0, bba, aaZ_0)$
$\vdash (q_1, ba, aZ_0)$
$\vdash (q_1, a, Z_0)$
$\vdash (q_2, Z_0)$

(iii) abb

$\delta(q_0, abb, Z_0) \vdash (q_0, bb, aZ_0)$
$\vdash (q_0, b, Z_0)$
$\vdash (q_0, \varepsilon, bZ_0)$ - Halted so rejected.

Q14 Design a PDA for palindrome (odd length also).

Soln: $L = \{010, 00100, 10101, \ldots\}$
101

Sir: Extra transition need to include for in Q12.

$\delta(q_0, 1, Z_0) = (q_2, Z_0)$
$\delta(q_0, 0, Z_0) = (q_2, Z_0)$ } If contains only one
ε0ε
ε1ε

$\delta(q_0, 1, 0) = (q_1, 0)$
$\delta(q_0, 1, 1) = (q_1, 1)$ } '1' is at middle element. 010
111

$\delta(q_0, 0, 0) = (q_1, 0)$
$\delta(q_0, 0, 1) = (q_1, 1)$ } 0 is at middle
000
101

Applications of Context free grammar.

1. Parses: To design programming language compilers & interpreters CFGs are very important. Usually languages are balanced Paranthesis which can be expressed using CFG Arithmetic and conditional expression along with various operators can also be easily expressed using CFG. Using CFG we can check syntax not semantics
validity of grammar
meaning of grammar.

2. YACC parser-generator Yet another compiler Compiler is a program in Unix which efficient parser can be generated. The input to this prog or command in CFG but represented using different notation.

3. Markup Language.
The most familiar markup languages HTML major function are: for creating links between documents and describing the look of a document. The structure of a part of HTML can be with some tags.

4. XML & document type definitions
XML describes the semantics of the text.

Scanned with CamScanner