21-7-19

Unit - 3

Context free Grammer (CFG)

The Context free grammer is a finite set of variables each of which represent a language.

The language represent by variable are described recurrecly in terms of each other & Primitive symbols called terminal.

The rules relating the variable are called Production.

non terminal: A ... Z

terminal : a ... z , +, -, *, /, 1 ... 9

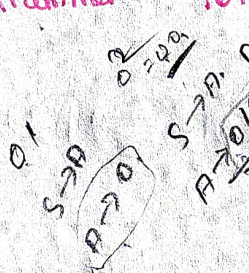
1) find the Context free Grammer for the Following language: $L = \{0^n / n=1\}$

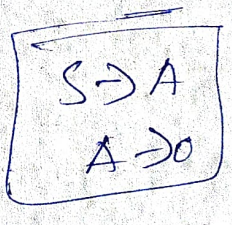
Soln:

$L = \{0^n / n=1\}$

$S \to B$

$B \to 0.$





2. find the CFG for the following language $L = \{0^n / n=1\}$

Soln:

$L = \{0^n / n \geq 1\}$

$S \to A$

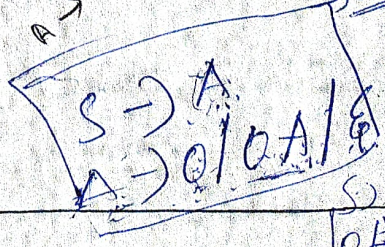
$A \to 0 / 0A / \epsilon$

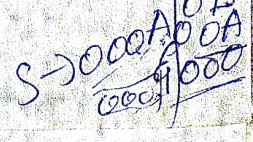
$L = 0$

$= 0^2 = 00$

$= 0^3 = 000$

$A \to 0 / 0A / \epsilon$





3. Find the CFG for the following language $L = \{0^n 1^n / n \geq 1\}$

Soln:

$S \rightarrow SB$

$A \rightarrow 0/0A/\epsilon$

$B \rightarrow 1/1B/\epsilon$

$S \rightarrow AB$
$A \rightarrow 0/0A/\epsilon$
$B \rightarrow 1/1B/\epsilon$

$S \rightarrow AB$ 01
$\rightarrow 0A1B$ 0011
$\rightarrow 00A11B$ 000111
$\rightarrow 000A111B$ 00001111

$A \rightarrow 0/0A/\epsilon$
$B \rightarrow 1/1B/\epsilon$

---

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

Soln:

$S \rightarrow A1A$

$A \rightarrow 0/0A/\epsilon$

$S \rightarrow A1A$ 010
$A \rightarrow 0/0A/\epsilon$ 00100
000100

Parse tree (or) derivation tree:

i) left most derivation
ii) Right " "

i) find left most & Right most derivation for the following Equation. $E \rightarrow E+E / E*E / (E) / a$ by the Production 'a+ a* a'

Soln:

left most derivation

$E \rightarrow E+E$
$\rightarrow E+E*E$
$\rightarrow a+E*E$
$\rightarrow a+a*E$
$\rightarrow a+a*a$

Right most derivation

$E \rightarrow E+E$
$\rightarrow E+E*E$
$\rightarrow E+E*a$
$\rightarrow E+a*a$
$\rightarrow a+a*a$

23-7-19

2) $(a+a)*a$

LMD ⇒ $E \to E*E$

$(E)*E$

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

$(a+a)*E$

$(a+a)*a$

RMD ⇒ $E*E$

$(E)*E$

$(E+E)*E$

~~$(a+a)*E$~~ $\to (E+E)*a$

$\to (E+a)*a$

$(a+a)*a$

---

(3) $E \to E+E \mid E*E \mid E-E \mid (E) \mid a$ $\quad (a*a)+(a-a)$

LED:

$E \to E+E$

$\Rightarrow (E)+E$

$\Rightarrow (E*E)+E$

$\Rightarrow (E*E)+(E)$

$\Rightarrow (E*E)+(E-E)$

$\Rightarrow (a*a)+(E-E)$

$\Rightarrow (a*a)+(a-E)$

$E \Rightarrow (a*a)+(a-a)$

RMD:

$E \to E+E$

$\Rightarrow E+(E)$

$\Rightarrow E+(E-E)$

$\Rightarrow (E)+(E-E)$

$\Rightarrow (E*E)+(E-E)$

$\Rightarrow (E*E)+(E-a)$

$\Rightarrow (E*E)+(a-a)$

$\Rightarrow (E*a)+(a-a)$

$E \Rightarrow (a*a)+(a-a)$

(4) $E \to E+E / E*E / \overset{E-E/}{\wedge}(E) / a$ (a+(a*a))

LED:

$E \to (E)\bullet$

$\to (E+E)$

$\to (E+(E))$

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

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

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

$\to (a+(a*a))$

RED:

$E \to (E)$

$\to (E+E)$

$\to (E+(E))$

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

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

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

$\to (a+(a*a))$

---

1. derive the string b* (b+ a11) using LMD derivation & RMD derivation from the Production $E \to I / E+E / E*E / (E)$

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

Soln:

LED:

$E \to E*E$

$\to E*(E)$

$\to E*(E+E)$

$\to I*(E+E)$

$\to b*(E+E)$

$\to b*(I+E)$

$\to b*(b+E)$

$\to b*(b+I)$

$\to b*(b+I_1)$

$\to b*(b+I_{11})$

$\to b*(b+a_{11})$

RED:

$E \to E*E$

$\to E*(E)$

$\to E*(E+E)$

$\to E*(E+I)$

$\to E*(E+I_1)$

$\to E*(E+I_{11})$

$\to E*(I+a_{11})$

$\to E*(b+a_{11})$

$\to I*(b+a_{11})$

$\to b*(b+a_{11})$

1-8-2019

Pairs tree

- $E \rightarrow E+E \mid E*E \mid (E) \mid a$

1) $a+a*a$

Soln:

LMD:

$E \rightarrow E+E$

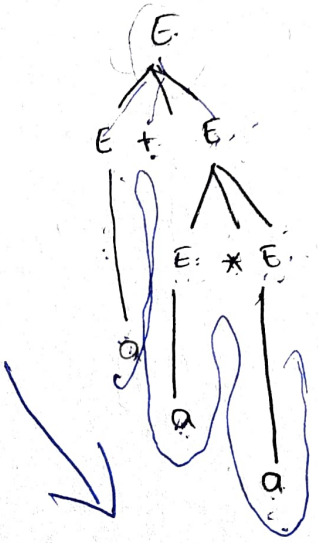
$\rightarrow E+E*E$

$\rightarrow a+E*E$

$\rightarrow a+a*E$

$E \rightarrow a+a*a$

LMD for Parse tree



RMD:

$E \rightarrow E*E$

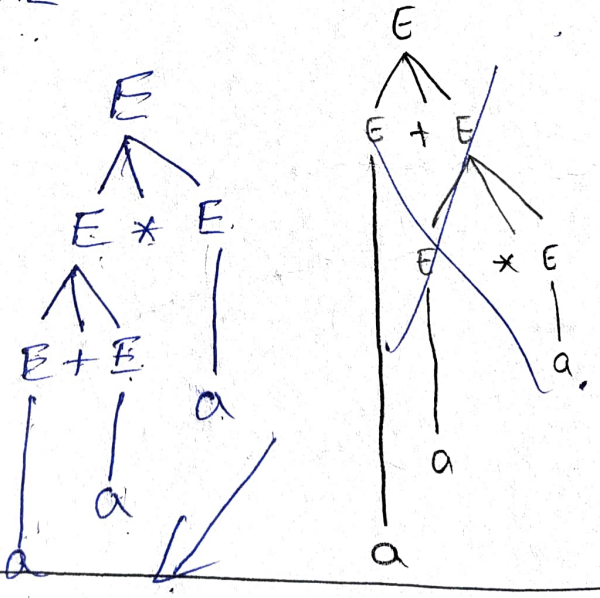
$\rightarrow E+E*E$

$\rightarrow E+E*a$

$\rightarrow E+a*a$

$\rightarrow a+a*a$

RMD for Parse tree



2) $(a+a)*a \Rightarrow E \rightarrow E+E \mid E*E \mid (E) \mid a$

LMD:

$E \Rightarrow E*E$

$\Rightarrow (E)*E$

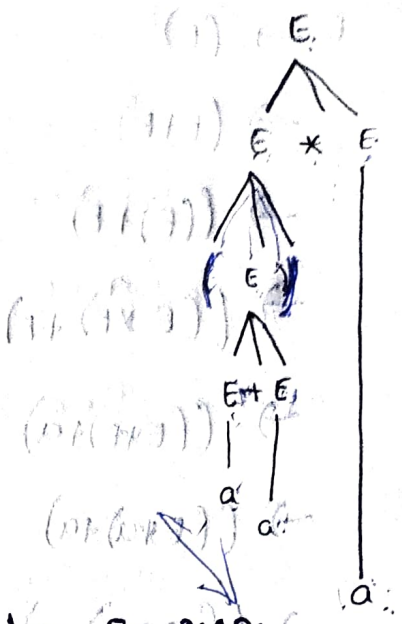
$\rightarrow (E+E)*E$

$\Rightarrow (a+E) * E$

$\Rightarrow (a+a) * E$

$\Rightarrow (a+a) * a$

Parse tree for LMD



RMD:

$E \rightarrow E * E$

$\rightarrow (E) * E$

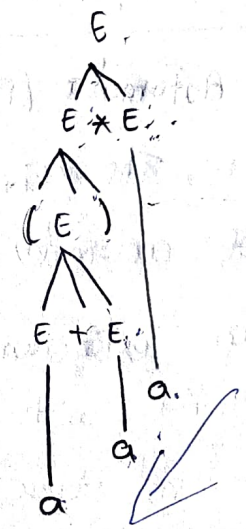
$\rightarrow (E+E) * E$

$\rightarrow (E+E) * a$

$\rightarrow (E+a) * a$

$\rightarrow (a+a) * a$

Parse tree for RMD:



---

3) $((a*a)+a) \Rightarrow E \rightarrow E+E \mid E*E \mid (E) \mid a$

LMD:

$E \rightarrow (E)$

$\rightarrow (E+E)$

$\rightarrow ((E)+E)$

$\rightarrow ((E*E)+E)$

$\rightarrow ((a*E)+E)$

$\rightarrow ((a*a)+E)$

$E \rightarrow ((a*a)+a)$

Parse tree for LMD

RMD :

$E \rightarrow (E)$

$\rightarrow (E+E)$

$\rightarrow ((E)+E)$

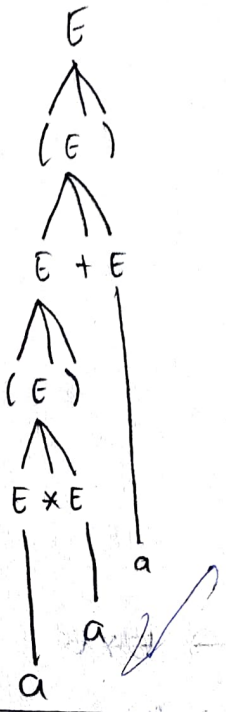
$\rightarrow ((E*E)+E)$

$\rightarrow ((E*E)+a)$

$\rightarrow ((E*a)+a)$

$\rightarrow ((a*a)+a)$

Parse tree for RMD



Push down Automata (PDA):

The Context free language is defined by a special type of automata namely Push down Automata. The Push down automata is an extension of non-deterministic finite automata with Epsilon (ε) transition with addition of the stack.

The stack is used to store the string of stack symbols

i) The stack is used to read the symbols Push & Pop only at the top of the stack.

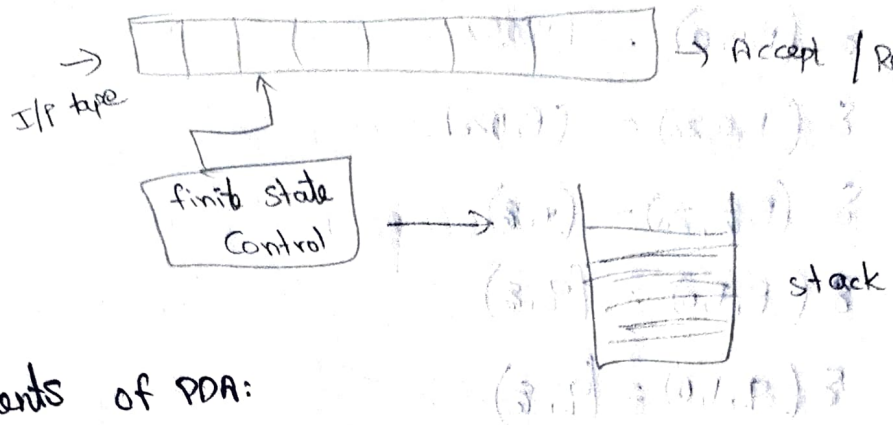
(ii) The Push down automata can remember infinite amount of the string information.

iii) The Push down automata can access the information on its stack in last in first out.

iv) The Pushdown automata can recognize only the context free language (CFL)

The Pushdown automata is more Powerful than finite automata which has finite memory.

PDA



Components of PDA:

The PDA Consists of the following Components

i) finite state Control (control unit)
ii) I/P tape
iii) Read head
iv) stack

formal definition of PDA:

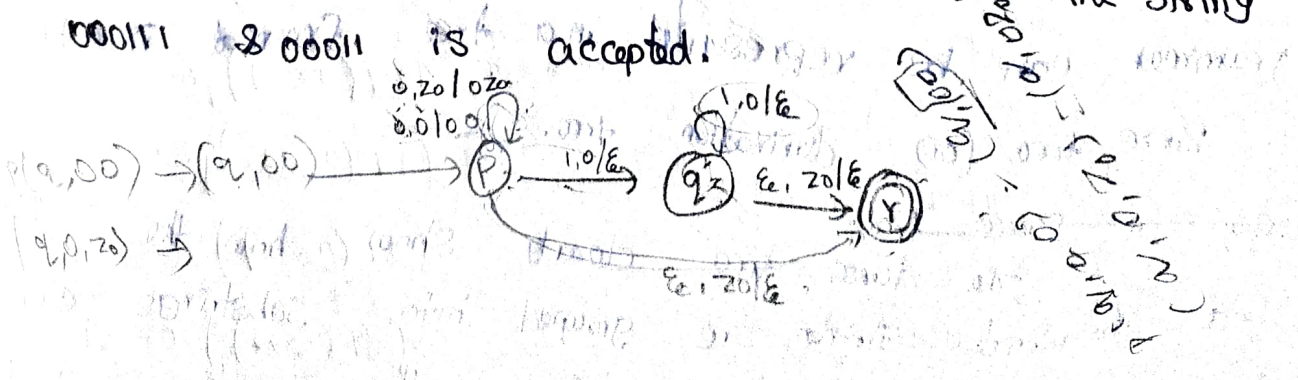
The PDA or non-deterministic PDA (NPDA) involves of components.

$\{Q, \Sigma, \tau, \delta, q_0, z_0, F\}$

where, Q → finite set of states
Σ - finite set of I/P Symbols
τ - finite state alphabet
δ - Transition function
q_0 - starting state of PDA
z_0 - starting symbol of stack (stack symbol)
F - final state of PDA (accepting state of PDA)

---

1) Consider the following PDA & find whether the string 000111 & 00011 is accepted.



$\delta(P,0,0) = (P,00)$

$\delta(P,0,z_0) = (P,0z_0)$

$\delta(P,\varepsilon,z_0) = (r,\varepsilon)$

$\delta(P,1,0) = (q,\varepsilon)$

$\delta(q,1,0) = (q,\varepsilon)$

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

00011

$\delta(P,000111,z_0)$

$= \delta(P,00111,0z_0)$

$= \delta(P,0111,00z_0)$

$= \delta(P,111,000z_0)$

$= \delta(P,11,00z_0)$

$= \delta(P,1,0z_0)$

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

$= (r,\varepsilon)$

00011 $\delta(P,00011,z_0)$

$= \delta(P,0011,0z_0)$

$= \delta(P,011,00z_0)$

$= \delta(P,11,000z_0)$

$= \delta(P,1,00z_0)$

$= \delta(q,\varepsilon,0z_0)$

$\delta(q,\varepsilon,0z_0)$ is not in the above transition function.

So the string 00011 is not accepted.

Parse tree:

The string that are derived from the context free grammer can be represented in a tree format is known as Parse tree (or) derivation tree.

~~Uses of Parse tree:~~

Uses of Parse tree:

The Parse tree clearly show how the symbols of a terminal strings are grouped into substrings each one which belongs to the language of one of this variable of a grammar

uses of Parse tree:

The Parse tree can be used for both derivation & re cursive language.

The Parse tree is used to find ambiguity in grammer & languages.

The Parse tree is used in Complier which facilitate the translation the source Program into executable code.

Ambiguity in Grammar & language:

If there is a grammar $G = (V, T, P, S)$ then the grammar G is ambigious, grammer is there is a string 'w' that can be derived two or more ways using some derivation method either LMD or RMD.

(or)

The grammar is ambigious, if & only if there exist atleast one string 'w' for which two or more different Parse tree exist by applying either LMD or RMD.

---

(2)

1) check whether the following grammar is ambigious.

$E \rightarrow E+E \mid E*E \mid E-E \mid a$

Soln:

Consider the string $w = a*a+a$

first LMD:

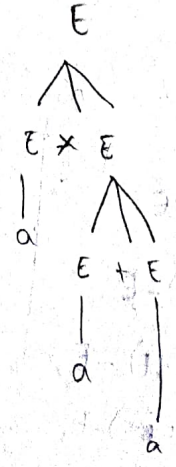
$E \rightarrow E*E$

$E \rightarrow E*E+E$

$E \rightarrow a*E+E$

$\rightarrow a*a+E$

$\rightarrow a*a+a$



Second LMD :

$E \rightarrow E+E$

$\rightarrow E*E+E$

$\rightarrow a*E+E$

$\rightarrow a*a+E$

$E \rightarrow a*a+a$

So, the string a*a+a is derived from two LMD derived from the above grammar.

So, the grammar $E \rightarrow E+E / E*E / E-E / a$ is ambigious grammar.

---

2) $w = a * (a+a) - a$ $\quad E \rightarrow E+E \mid E*E \mid E-E \mid (E) \mid a$

first LMD:

$E \rightarrow E*E$

$\rightarrow E*E-E$

$\rightarrow E*(E)-E$

$\rightarrow E*(E+E)-E$

$\rightarrow a*(E+E)-E$

$\rightarrow a*(a+E)-E$

$\rightarrow a*(a+a)-E$

$E \rightarrow a*(a+a)-a$

Second LMD:

$E \rightarrow E-E$

$\rightarrow E*E-E$

$E*(E)-E$

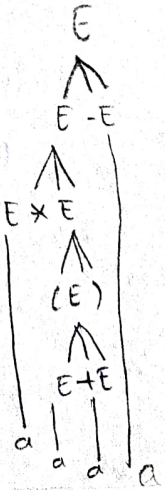
$E*(E+E)-E$

$a*(E+E)-E$

$a*(a+E)-E$

$a*(a+a)-E$

$a*(a+a)-a$



So the string a*(a+a)-a is derived from two left most derivation derived from the above grammer

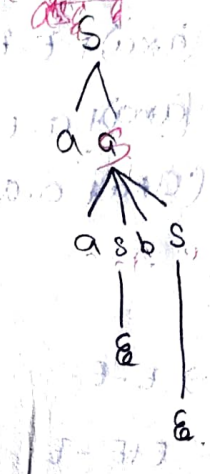
So, the grammer E → E+E / E*E / E-E / (E) / a is ambigious grammer.

---

3) S = as / asbs / ε 'aab'

First LMD

S → as
→ aaSbs
→ aaεbs
→ aabs
→ aabε
S → aab



Second LMD:

S → asbs
→ aasbs
→ aaεbs
→ aabs
→ aabε
S → aab

So the string 'aab' is derived from two LMD derived from the above grammer.

So, the grammar S → as / bsas / ε is ambigious grammer.

Removing Ambiguity from the Gramma of unambiguity:

1. Check whether the following CFG is ambigious and remove the ambigious & make the grammar has unambigious.

E → E*E / E+E / E-E / (E) / a

first CMD: Let us consider the string (a*a)+a-a

$E \to E+E$

$\to (E)+E$

$\to (E*E)+E$

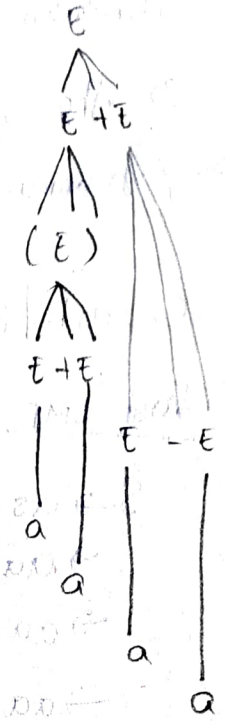
$\to (E*E)+E-E$

$\to (a*E)+E-E$

$\to (a*a)+E-E$

$\to (a*a)+a-E$

$\to (a*a)+a-a$



Second CMD:

$E \to E-E$

$\to E+E-E$

$\to (E)+E-E$

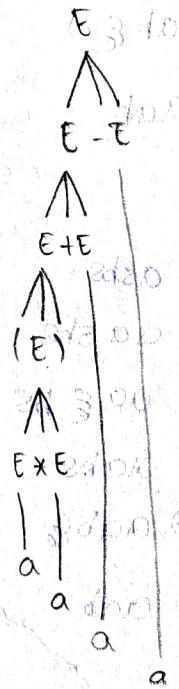
$\to (E*E)+E-E$

$\to (a*E)+E-E$

$\to (a*a)+E-E$

$\to (a*a)+a-E$

$\to (a*a)+a-a$



(a*a)+a-a

we have a two Parse tree for a single string

So, the above grammar is ambigious grammar.

(a+a)+a-a

Un ambigious:

$E \to E*E \mid E+E \mid E-E \mid E/E \mid (E) \mid a$

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

$T \to E-X \mid E/X$

$X \to (E) \mid a$

(a*a)+a-a

first CMD:

$(a*a)+a-a$

$E \to E+T$

$\to E+E-X$

$E \to E+E-a$

So, the string $(a*a)+a-a$ is not derived from the above grammar

So, the above grammar is unambiguous.

---

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

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

Let as consider the string $a*b+boo$

$E \to E+E$
$\to I+E$
$\to a+E$
$\to a+I$
$\to a+Ib$
$\to a+Ibb$
$\to a+abb$

First LMD:

$E \to E*E$

$\to E*E+E$

$\to I*E+E$

$\to a*E+E$

$\to a*I+E$

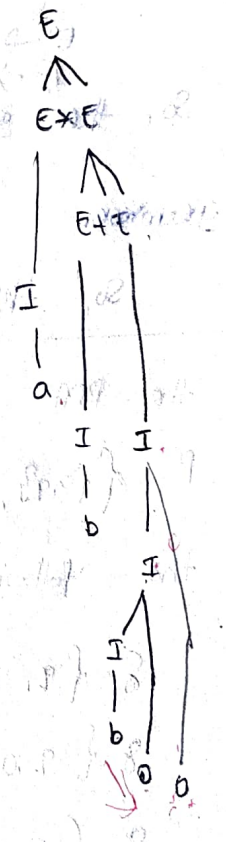
$\to a*b+E$

$\to a*b+I$

$\to a*b+I0$

$\to a*b+I00$

$\to a*b+boo$



Second LMD:

$E \to E+E$

$\to E*E+E$

$\to I*E+E$

$\to a*E+E$

$\to a*I+E$

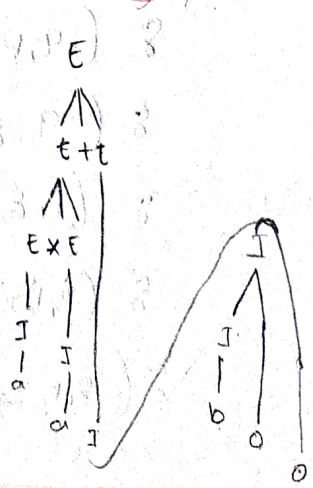
$\to a*b+E$

$\to a*b+I$

$\to a*b+I0$

$\to a*b+I00$

$\to a*b+boo$



So, the string a+b+boo is derived from two LMD.

So, the above grammar is ambigious grammar.

Unambigious:

$E \rightarrow I \mid E*E \mid E+E \mid (E)$

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

$E \rightarrow X \mid E*E$

$X \rightarrow E+T \mid (E)$

$T \rightarrow Za \mid Zb$

$Z \rightarrow Z0 \mid Z1 \mid a \mid b$

first LMD: a*b+boo

$E \rightarrow E*E$
$\rightarrow X*E$
$\rightarrow E+T*E$
$\rightarrow E+Z_1*E$
$\rightarrow E+ba*E$

$E \rightarrow E*X$
$E \rightarrow *$

So, the string a*b+boo is not derived from the above grammar.

So, the above grammar is unambigious.

---

1) Suppose the PDA

$P = \{\{p,q\}, \{0,1\}, \{z_0, x\}, \delta, q_0, z_0, \{p\}\}$

assume the following transition function.

$\delta\{q_1, 0, z_0) = \{q_1, xz_0)$

$\delta\{q_1, 0, x) = (q_1, xx)$

$\delta(q_1, 1, x) = (q, x)$

$\delta(q, \varepsilon, x) = (p, \varepsilon)$

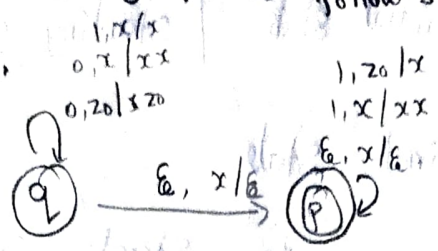
$\delta(p, \varepsilon, x) = (p, \varepsilon)$

$\delta(p, 1, x) = (p, xx)$

$\delta(p, 1, z_0) = (p, x)$

when the I/P string W as follows $w = 0011$, $w = 010$ is accepted or not.

Soln:



$W = 0011$

$\delta(q, 0011, z_0) \rightarrow \delta(q, 011, xz_0)$

$= \delta(q, 011, 0z_0) \rightarrow \delta(q, 11, xx)$

$= \delta(q, 11, 00z_0) \rightarrow \delta(p, 1, x)$

$= \delta(p, 1, 0z_0) \rightarrow \delta(p, \varepsilon, xx)$

$= \delta(p, \varepsilon, z_0) \rightarrow \delta(p, \varepsilon)$

$\therefore$ p is a final state $W = 0011$ is accepted.

$W = 010$

$\delta(q, 010, z_0) \rightarrow \delta(q, 010, z_0)$

$\delta(q, 10, 0z_0) \rightarrow \delta(q, 10, xz_0)$

$\delta(p, 0, z_0) \rightarrow \delta(p, 0, x)$

$(p, 0, z_0)$ is not in the transistion function.

---

7) design PDA to accept the language $L = \{a^n b^n / n \geq 1\}$ accepting by final state.

(or)

design PDA for the set of all string $\{a, b\}$ with equal no. of a's & equal no. of b's

Soln:

Here, the string Contains equal no of a's & b's it also have the empty string.

let q_0 ⟶ initial state / starting state

q_2 ⟶ final state

z_0 ⟶ top of the stack Symbol.

The idea to design this PDA is that when over we read 'a', initially push all a's to the stack.

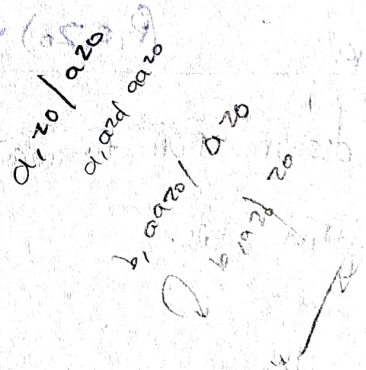
we read all b's we delete 'a' from the stack. that is for each 'a' in the stack. we delete 'a' from the stack for read each 'b'

When we reach the empty string, then the string Contains z_0 Symbols.

after reading the entire string, we delete the stack symbols z_0 & reach the final state.

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

aabbcc



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

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

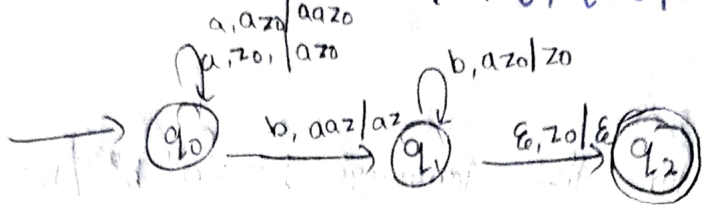
$(q_0, b, a a z_0) = (q_1, a z_0)$

$(q_1, b, a z_0) = (q_1, z_0)$

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

PDA:

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



Consider the I/p string 'aabb' & 'aab'

aabb

$(q_0, aabb, z_0)$

$= (q_0, abb, az_0)$

$= (q_0, bb, aaz_0)$

$= (q_1, b, az_0)$

$= (q_1, \varepsilon, z_0)$

$\rightarrow (q_2, \varepsilon)$

q_2 is final state so aabb is accepted.

'aab'

(q_0, aab, z_0)

$= (q_0, ab, az_0)$

$= (q_0, ab, aaz_0)$

$= (q_1, \varepsilon, az_0)$

(q_1, ε, az_0) is not in the PDA of above transition function.

~~q_1 is not final state, so 'aab'~~ So, 'aab' is not accepted.

2) design PDA to accepted the language $L = \{a^n b^n / n \geq 1\}$

3) design PDA for the language $L = \{a^n b^{2n} / n \geq 1\}$ is accepting by final state

(or)

design PDA with the set of String with (two b's) twice many b's than a's with as the starting String.

Soln:

the idea to design PDA is that when we read single 'a' we push two a's on the state

When we read 'b' we Pop each 'a' until top of the stack Z_0, we reach the final state

let $q_0 \rightarrow$ starting state

$q_2 \rightarrow$ final state

$Z_0 \rightarrow$ top of the Stack Symbol.

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

aabbbb

$(q_0, a, Z_0) = (q_0, aaZ_0)$

$(q_0, a, aaZ_0) = (q_0, aaaaZ_0)$

$(q_0, b, aaaaz_0) = (q_1, aaaz_0)$

$(q_1, b, aaaz_0) = (q_1, aaz_0)$

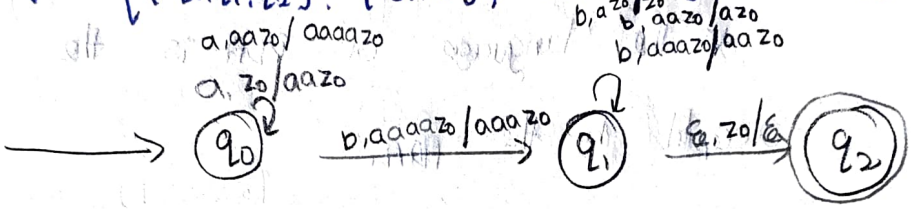
$(q_1, b, aaz_0) = (q_1, az_0)$

$(q_1, b, az_0) = (q_1, z_0)$

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

PDA:

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



Consider the I/P string 'aabbbb' & aabb

aabbbb

$(q_0, aabbbb, z_0)$

$= (q_0, abbbb, az_0)$

$= (q_0, bbbb, aaz_0)$

$= (q_1, bbb, az_0)$

$= (q_1, bb, z_0)$

$= (q_2, b)$

q_2 is final state & 'aabb' is accepted.

let 'aab'

$\rightarrow (q_0, aab, z_0)$

$\rightarrow (q_0, ab, az_0)$

(q_0, b, aaz_0)

(q_1, ϵ, az_0)

$\therefore (q_1, \epsilon, az_0)$ is not in the PDA of above transition function. So 'aab' is not accepted.

18-8-15

language to PDA / language of PDA:

* The language of PDA is the set of all strings accepted by the PDA.

* Generally, the string is accepted by PDA after consuming to input, the PDA should either final state.

* Else after reading the I/P, stack should be empty, then the language is accepted by PDA.

* So, two methods on accepting a string in the PDA as follows.

1) acceptance by final state
2. acceptance by empty stack

1. Acceptance by final state

let $P = \{Q, \Sigma, \Gamma, \delta, \{q_0\}, z_0, F\}$

be a PDA then the language $L(P)$ is accepted by

Final State

$$L(P) = \{w / q_0, w, Z_0, T_P, (q, \varepsilon, \alpha)\}$$

q - final state

α - any stack symbol

acceptance by empty stack:

let $P = \{Q, \Sigma, \Gamma, \delta, \{q_0\}, Z_0, F\}$

be a PDA then the language $L(P)$ is accepted by the empty stack.

$$L(P) = \{w / q_0, w, Z_0, T_P, (q, \varepsilon, \alpha)\}$$

Problem on Converting (CFG to) PDA:

1) 13m (5) Construct PDA for the following grammer;

$E \rightarrow E+E \mid E*E \mid a$

Soln:

non terminal $\Rightarrow E$

$$\delta(q, E) = \{(q, E+E), (q, E*E), (q, a)\}$$

Terminal $\Rightarrow +, *, a$

$$\delta\{q, +, +\} = (q, \varepsilon)$$

$$\delta\{q, *, *\} = (q, \varepsilon)$$

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

Consider the I/P string $w = 'a*a+a'$

transition function:

$\rightarrow \delta(q, a*a+a, E)$

$\rightarrow \delta(q, a*a+a, E*E)$

$\rightarrow \delta(q, a*a+a, a*E)$

$\rightarrow \delta(q, *a+a, *E)$

$\rightarrow \delta(q, a+a, E)$

$\rightarrow \delta(q, a+a, E+E)$

$\rightarrow \delta(q, a+a, a+E)$

$\rightarrow \delta(q, +a, +E)$

$\rightarrow \delta(q, a, E)$

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

$\rightarrow \delta(q, \epsilon, \epsilon)$

Thus the CFG accept the string 'a*a+a' and it's accepted by PDA in empty stack.

. Constact PDA for following grammer

$E \rightarrow E*E \mid E+E \mid E-E \mid E/E \mid I$

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

Terminal → a...z, +, -, *, / 0...9

non-terminal → A...Z

soln:

non terminal: E, I

$\delta(q, E) = \{(q, E*E)\ (q, E+E)\ (q, E-E)\ (q, E/E),\ (q, I)\}$

$$\delta(q,I) = \{ (q,Id), (q,Ib), (q,I0), (q,I.), (q,a), (q,b) \}$$

terminal: *, +, -, /, a, b, 0, 1

transition function:

- $\delta(q,*,*) = (q,\varepsilon)$
- $\delta(q,+,+) = (q,\varepsilon)$
- $\delta(q,-,-) = (q,\varepsilon)$
- $\delta(q,/,/) = (q,\varepsilon)$
- $\delta(q,a,a) = (q,\varepsilon)$
- $\delta(q,b,b) = (q,\varepsilon)$
- $\delta(q,0,0) = (q,\varepsilon)$
- $\delta(q,1,1) = (q,\varepsilon)$

Consider the I/P string 'a*a+b11'

transition function:

$= \delta(q, a*a+b11, \varepsilon)$

$= \delta(q, a*a+b11, \varepsilon*\varepsilon)$

$\delta(q, a*a+b11, I*\varepsilon)$

$\delta(q, a*a+b11, a*\varepsilon)$

$\delta(q, *a+b11, *\varepsilon)$

$\delta(q, a+b11, \varepsilon)$

$\delta(q, a+b11, \varepsilon+\varepsilon)$

$\delta(q, a+b||, I+E)$

$\delta(q, a+b||, a+E)$

$\delta(q, +b||, +E)$

$\delta(q, b||, E)$

$\delta(q, b||, I)$

$\delta(q, b||, I,)$

$\delta(q, b||, I||)$

$\delta(q, b||, b||)$

$\delta(q, ||, ||)$

$\delta(q, 1, 1)$

$\delta(q, \varepsilon, \varepsilon)$ (or) $\delta(q, \varepsilon)$

Thus CFG accept the string 'a* a+b||' & accepted by PDA in empty Stack.

2) Construct PDA

$S \rightarrow aAA$

$A \rightarrow aS/bS/a$

3) $S \rightarrow AA/0$

$A \rightarrow S/1$