Push Down Automata (PDA)

⇒ Using pumping lemma for regular expression, we come to know that there are certain Languages like $L = \{a^n b^n : n>0\}$ doesn't belong to the class of FA. i.e. such type of Languages are not recognized by FA.

⇒ This is because, FA is a memory less machine and to process the Language like above, FA must have to remember the total no. of a's inorder to process the same number of b's.

⇒ To process such type of Languages, we have machine which can remember the number of a's such that it can match exact number of b's with respect to a's. Such machine is called pushdown Automata (PDA).

⇒ In pushdown Automata (PDA), stack is used in the form of memory. Since stack is also called pushdown store, it is called pushdown automata.

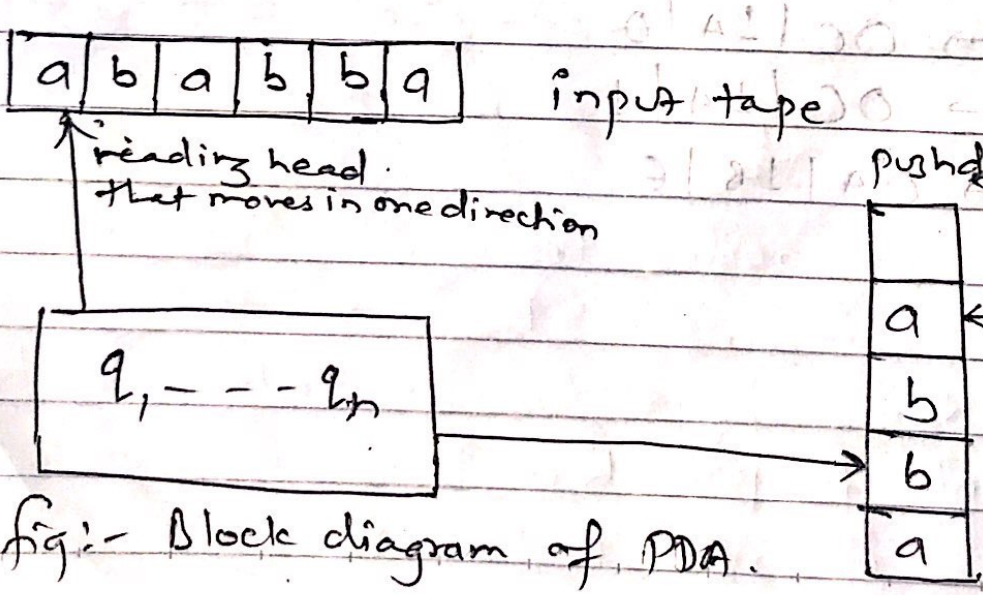


fig:- Block diagram of PDA.

Date: Page:

- PDA is an abstract (idealized) machine determined by following three things.
  - Input tape
  - finite control
  - stack.

- Each moves of the machine is determined by three things.
  - The current state
  - Next input symbol
  - Symbol on the top of stack.
- The move consists of;
  - changing state / staying on same state.
  - Replacing the stack top by string of zero or more symbols.
- Popping the top symbol off the stack means replacing it by ϵ.
- Pushing Y on the stack means replacing stack's top. say X, by YX. The single move of the machine contains only one stack operation either push or pop.

formal Definition :-

A PDA can be defined by seven tuples.

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

Where

Q = finite set of states

Σ = finite set of input symbols / alphabets.

δ = transition function
q_0 = initial state.
F = finite set of final states.
Γ = Symbol of stack / finite set of stack alphabet.
Z_0 = initial stack symbol, $Z_0 \in \Gamma$.

Moves of PDA :-

- Each instant provided by the transition function of a PDA is called its move. The move of a PDA is defined as :-

$$\delta(q, X, Y) \longrightarrow (P, Z)$$

Where,

q — current state
X — Symbol of the input tape under reading head.
Y — top of stack
P — next state
Z — new top of stack that replaces Y.

- This move tells that, whenever we are in state q, if we read X from the input tape and Y is the current top of stack then state is changed to P and top of stack Y is replaced by Z, which becomes new top of stack.

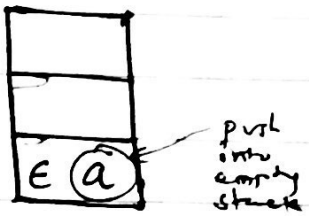
also $\delta(q, a, b) \rightarrow (p, a)$
replace tos (b) by a

Date: Page:

- To push an input symbol in the empty stack, we use the transition as;

$$\delta(q, a, \epsilon) \longrightarrow (p, a)$$

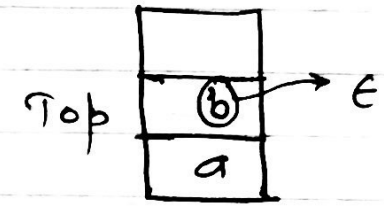
Pushes 'a' to the stack



- To pop an input symbol from stack we use transition as;

$$\delta(q, a, b) \longrightarrow (p, \epsilon)$$

pops 'b' from the stack.



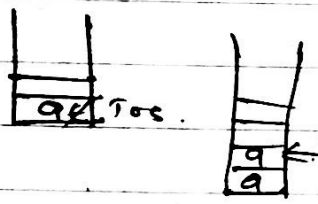
Q Design a PDA for the language.

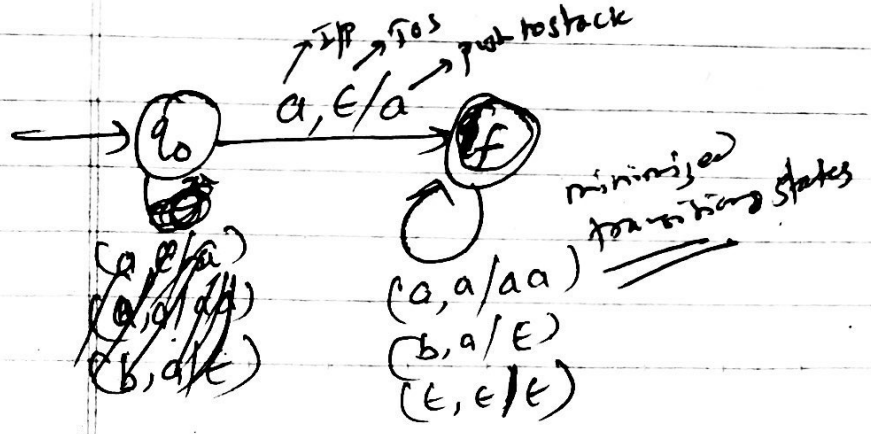
$$L = \{a^n b^n : n > 0\}$$

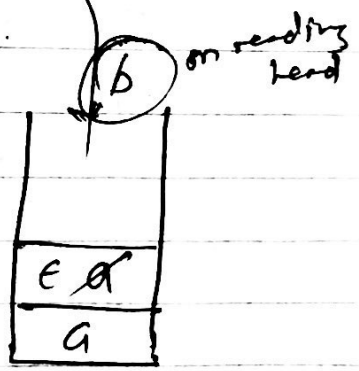
Soln: Let $M = (Q, \Sigma, \delta, q_0, F, \Gamma, Z_0)$ be a PDA.

δ can be defined as;

① $\delta(q_0, a, \epsilon) \longrightarrow (f, a)$
② $\delta(f, a, a) \longrightarrow (f, aa)$
③ $\delta(f, b, a) \longrightarrow (f, \epsilon)$
④ $\delta(f, \epsilon, \epsilon) \longrightarrow (f, \epsilon)$







for 'aaabbb' string

state	unread i/p	stack content	Transition used
q_0	aaabbb	ϵ	—
f	aabbb	a	(1)
f	aabbb	aa	(2)
f	abbb	aaa	(2)
f	bbb	aa	(3)
f	bb	a	(3)
f	b	a	(3)
f	ϵ	ϵ	(3)
f	—	ϵ	(4)

Since, the stack becomes empty after the termination of string, the language is accepted by PDA.

OR

① $\delta(q_0, \epsilon, \epsilon) \longrightarrow (q_1, Z_0)$

This step initializes the stack to indicate the bottom of stack.

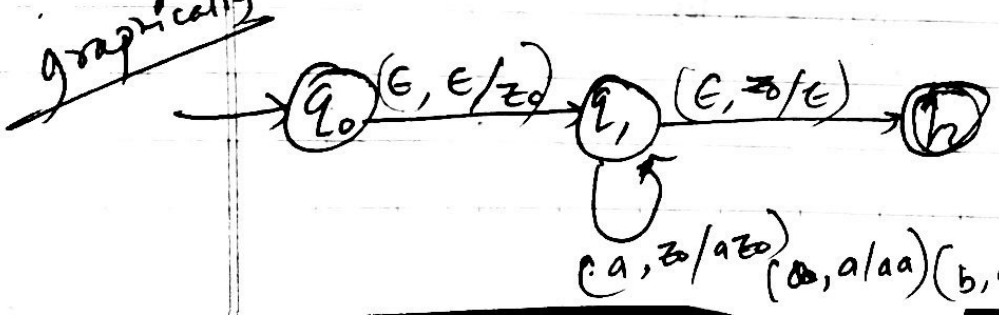
② $\delta(q_1, a, Z_0) \longrightarrow (q_1, aZ_0)$

③ $\delta(q_1, a, a) \longrightarrow (q_1, aa)$

④ $\delta(q_1, b, a) \longrightarrow (q_1, \epsilon)$

⑤ $\delta(q_1, \epsilon, Z_0) \longrightarrow (q_2, \epsilon)$

graphically



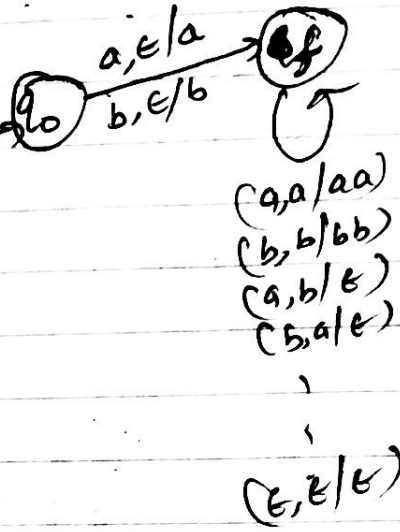
Date: Page:

Design a PDA for language

$L = \{$ Consisting all the strings of a's and b's over $\Sigma = \{a, b\}$, having equal no. of a's & b's $\}$.

Soln:

(1) $\delta(q_0, a, \epsilon) \longrightarrow (f, a)$
(2) $\delta(q_0, b, \epsilon) \longrightarrow (f, b)$
(3) $\delta(f, a, a) \longrightarrow (f, aa)$
(4) $\delta(f, b, b) \longrightarrow (f, bb)$
(5) $\delta(f, a, b) \longrightarrow (f, \epsilon)$
(6) $\delta(f, b, a) \longrightarrow (f, \epsilon)$
(7) $\delta(f, a, \epsilon) \longrightarrow (f, a)$
(8) $\delta(f, b, \epsilon) \longrightarrow (f, b)$
(9) $\delta(f, \epsilon, \epsilon) \longrightarrow (f, \epsilon)$

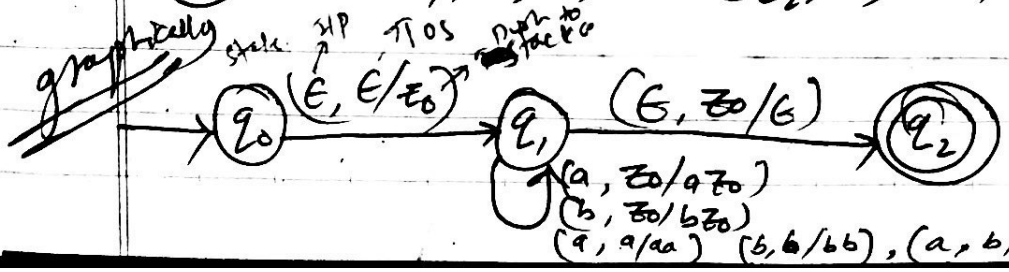


OR

(1) $\delta(q_0, \epsilon, \epsilon) \longrightarrow (q_1, Z_0)$ // initialize the stack to indicate the bottom of stack.

(2) $\delta(q_1, a, Z_0) \longrightarrow (q_1, aZ_0)$
(3) $\delta(q_1, b, Z_0) \longrightarrow (q_1, bZ_0)$
(4) $\delta(q_1, a, a) \longrightarrow (q_1, aa)$
(5) $\delta(q_1, b, b) \longrightarrow (q_1, bb)$
(6) $\delta(q_1, a, b) \longrightarrow (q_1, \epsilon)$
(7) $\delta(q_1, b, a) \longrightarrow (q_1, \epsilon)$
(8) $\delta(q_1, \epsilon, Z_0) \longrightarrow (q_2, \epsilon)$ // indicate the acceptance of string.
(9) $\delta(q_1, a, \epsilon) \longrightarrow (q_1, a)$
(10) $\delta(q_1, b, \epsilon) \longrightarrow (q_1, b)$

graphically



# trace for w = aabbbaab also
w = ababa

Q/ NPDA $L = \{ww^R \mid w \in \{a,b\}\}$

eg
a a
b b
ab ba
ba ab
w w^R

Rewrite.

a ∈ a
b ∈ b
ab ∈ ba
ba ∈ ab
← left string → right string

we need a separator such that we can identify the first half & second half

Sol'n: δ consists of:
A set to push w on the stack.

① $\delta(q_0, a, a) \longrightarrow \{(q_0, aa)\}$ a on top of a (TOS)
② $\delta(q_0, b, b) \longrightarrow \{(q_0, bb)\}$ b on top of b
③ $\delta(q_0, a, b) \longrightarrow \{(q_0, ab)\}$ a on top of b
④ $\delta(q_0, b, a) \longrightarrow \{(q_0, ba)\}$
⑤ $\delta(q_0, a, Z_0) \longrightarrow \{(q_0, aZ_0)\}$
⑥ $\delta(q_0, b, Z_0) \longrightarrow \{(q_0, bZ_0)\}$

Date: Page:

⇒ A set to guess the middle of stack where PDA switches from state q_0 to q_1 without reading anything.

(7) $\delta(q_0, \epsilon, a) \longrightarrow \{(q_1, a)\}$

(8) $\delta(q_0, \epsilon, b) \longrightarrow \{(q_1, b)\}$

⇒ A set to match w^R against the contents of stack.

(9) $\delta(q_1, a, a) \longrightarrow \{(q_1, \epsilon)\}$

(10) $\delta(q_1, b, b) \longrightarrow \{(q_1, \epsilon)\}$

if TOS & I/p match then pop TOS

⇒ and finally.

(11) $\delta(q_1, \epsilon, Z_0) \longrightarrow \{(q_f, Z_0)\}$ (q_2)

(12) also $\delta(q_0, \epsilon, \epsilon) \longrightarrow (q_0, Z_0)$

Date: Page:

Instantaneous Description (ID) of PDA:-

⇒ Any configuration of a PDA at a particular instant can be described by a triplet (q, w, γ)

where

q - is the present state of PDA.

w - is the remaining input (yet to be read by PDA)

γ - is the stack contents.

diagram

⇒ Such a description by triplet is called ID of PDA.

let $P = (Q, \Sigma, \delta, q_0, F, \Gamma, Z_0)$ be a PDA

Then we define a relation $\vdash$, "yields" as

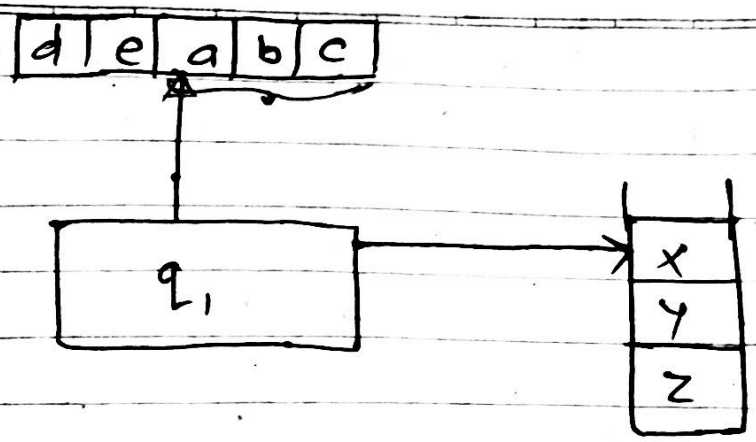
$$(q, aw, z\alpha) \vdash (p, w, \beta\alpha)$$

ID — from this configuration PDA will reach to this configuration in only one move.

ID — read 'a' from I/p tape, current TOS is z after one move PDA reaches to next configuration.

~~$\delta(q, a, z)$ contains (p, β)~~.

- This moves reflects the idea that, by consuming "a" from the input tape, and replacing z on the TOS by β, we can go from state q to state p.



(q_1, abc, XYZ) is the instantaneous description of above figure.

eg:
ID of PDA

=> for the PDA described earlier accepting language of equal a's & b's the accepting sequence of ID's for string abba can be shown as;

$(q_0, abba, \epsilon) \vdash (q_1, abba, Z_0)$
$\vdash (q_1, bba, aZ_0)$
$\vdash (q_1, ba, Z_0)$
$\vdash (q_1, a, bZ_0)$
$\vdash (q_1, \epsilon, Z_0)$
$\vdash (q_2, \epsilon, \epsilon)$ Accepted

Q Design a PDA accepting language
$L = \{ wcw^R \mid w \in (0+1)^* \}$.

Soln:

(1) $\delta(q_0, \epsilon, \epsilon) \longrightarrow (q_0, Z_0)$

(2) $\delta(q_0, 0, Z_0) \longrightarrow (q_0, 0Z_0)$

(3) $\delta(q_0, 1, Z_0) \longrightarrow (q_0, 1Z_0)$

(4) $\delta(q_0, 0, 0) \longrightarrow (q_0, 00)$

(5) $\delta(q_0, 1, 1) \longrightarrow (q_0, 11)$

(6) $\delta(q_0, 0, 1) \longrightarrow (q_0, 01)$

(7) $\delta(q_0, 1, 0) \longrightarrow (q_0, 10)$

(8) $\delta(q_0, c, 0) \longrightarrow (q_1, 0)$ → only state change, no change tos

Change the state when center is reached

(9) $\delta(q_0, c, 1) \longrightarrow (q_1, 1)$ → only state change, no change on tos

(10) $\delta(q_0, c, Z_0) \longrightarrow (q_1, Z_0)$ case only one string ε

(11) $\delta(q_1, 0, 0) \longrightarrow (q_1, \epsilon)$

(12) $\delta(q_1, 1, 1) \longrightarrow (q_1, \epsilon)$

(13) $\delta(q_1, \epsilon, Z_0) \longrightarrow (q_f, \epsilon)$

q_2

* String processing for w = "1 ... "

eg $w = aabbbb$

State	Unread i/p	stack contents	transition used
q	aabbbb	ϵ	—
f	abbbb	a	(1)
f	bbbb	aa	(2)
f	bbb	ba	(3)
f	bb	a	(4)
f	b	b	(3)
f	ϵ	ϵ	(4)
f	—	ϵ	(5)

'a' replaced by 'b'

Language of PDA:-

⇒ We can define acceptance of any string by a PDA in 2 ways.

(1) Acceptance by final state:-

— Given a PDA M, the language accepted by final state $L(M)$ is

$$\{w \mid (q, w, z_0) \xrightarrow{*} (P, \epsilon, \gamma)\}$$

where.

$P \in F$ and $\gamma \in \Gamma^*$

Date: Page:

2) Acceptance by empty stack :-

- Given a PDA M, the language accepted by empty stack, $L(M)$ is

$$\{w / (q, w, z_0) \vdash^{*} (p, \epsilon, \epsilon)\}$$

where,

$p \in Q$

Q. $L = \{ ww^R \mid w \in (a+b)^* \}$

$w = ab$ $w^R = ba$ $ww^R = abba$

$w = abbb$ $w^R = bbba$ $ww^R = abbbbbba$

$w = aba$ $w^R = aba$ $ww^R = abaaba$

we don't have middle marker
we cannot count the length.

one way:

a|bb|a

middle point already reached

abb|bb|bba

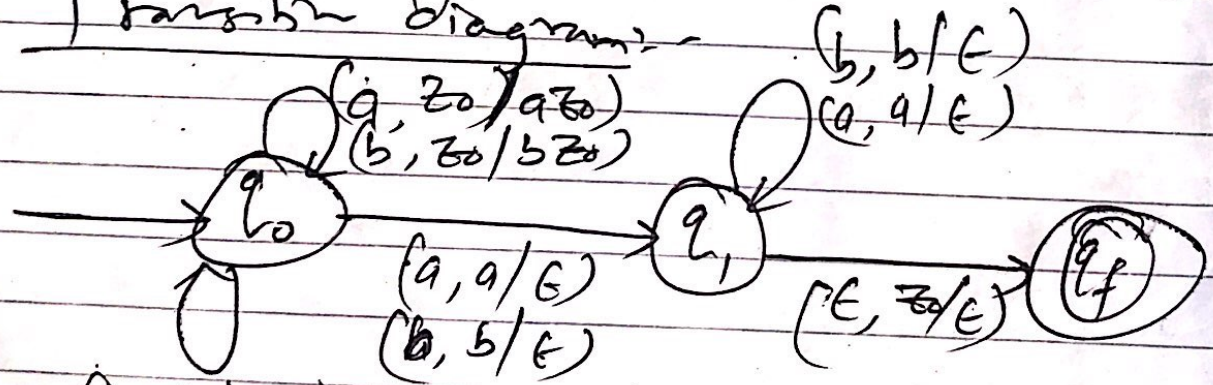
we can only guess middle point has reached.

there can be more middle points

abba|ba

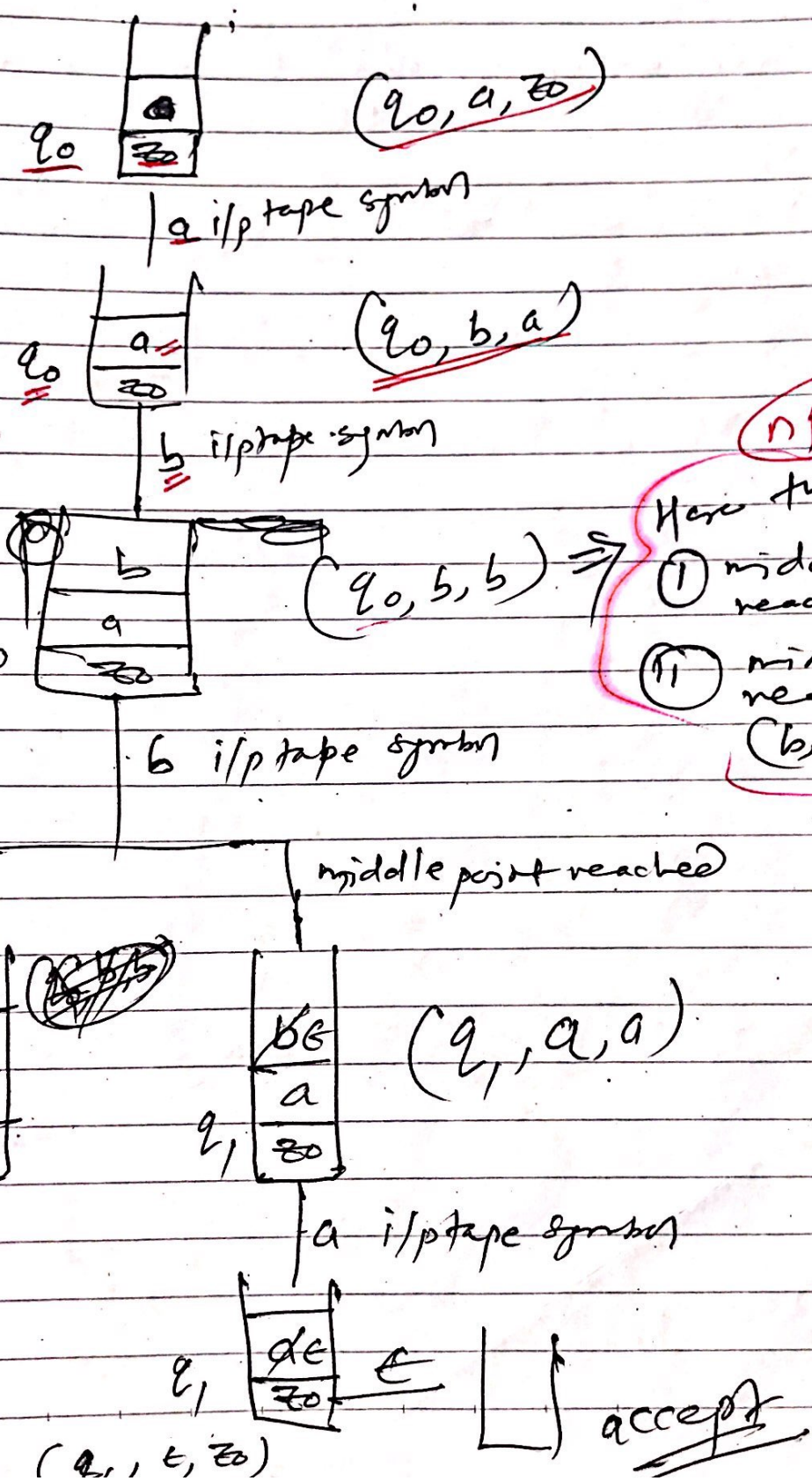
- we have to identify correct middle point

Transition diagram:-



Date: Page:

eg $w = abba$



npda

Here two transitions

(i) middle point not reached (b, bb)

(ii) middle point reached (b, ϵ)

Equivalence of CFG & PDA :-

⇒ CFG to PDA conversion :-

⇒ To convert a given CFG to it's equivalent PDA, it is needed to convert all the production rules of the given CFG into their equivalent transition functions.

⇒ Non-terminals of the given CFG are pushed to the stack and terminal symbols are popped from the stack.

⇒ We can define pushdown automata for the CFG with two steps P and q, where P being start state.

⇒ Here, idea is that, the stack symbol initially is supposed to be ϵ and PDA starts with state P and on reading ϵ symbol, insert start symbol S of CFG into stack.

⇒ PDA changes the state to q. Then all transition occur in state q.

ie. PDA can be defined as,

$M = (\{P, q\}, T, V \cup T, \delta, P, \{q\})$;

Stack top is ϵ,

then δ can be defined as,

$\delta(P, \epsilon, \epsilon) = \{(q, S) \mid S \text{ is start symbol of grammar G}\}$

$\delta(q, \epsilon, A) = \{(q, \alpha) \mid A \to \alpha \text{ is a production P of G}\}$

$\delta(q, a, a) = \{(q, \epsilon) \text{ for all } a \in \Sigma\}$

eg:- Convert the following CFG to its equivalent PDA.

$S \rightarrow 0S1 \mid 0AA \mid 1BB$
$A \rightarrow 1A \mid 0$
$B \rightarrow 0B \mid 1$

Check whether $w = 0010101$ is accepted by the PDA.

Soln:

Let $M = (Q, \Sigma, \delta, q_0, F, T)$ where
$Q = \{q, f\}$
$\Sigma = \{0, 1\}$

δ consists of

(1) $\delta(q, \epsilon, \epsilon) \longrightarrow (f, S)$
(2) $\delta(f, \epsilon, S) \longrightarrow (f, 0S1), (f, 0AA), (f, 1BB)$
(3) $\delta(f, \epsilon, A) \longrightarrow (f, 1A), (f, 0)$
(4) $\delta(f, \epsilon, B) \longrightarrow (f, 0B), (f, 1)$
(5) $\delta(f, 0, 0) \longrightarrow (f, \epsilon)$
(6) $\delta(f, 1, 1) \longrightarrow (f, \epsilon)$
(7) $\delta(f, \epsilon, \epsilon) \longrightarrow (f, \epsilon)$

$q_0 = q$

$F = f$

$\Gamma = \{0, 1, S, A, B\}$

State	Unread i/p	stack content	transition used
q	∈0010101	∈	—
f	∈0010101	S	(1)
f	0010101	0S1	(2)
f	∈010101	S1	(5)
f	0010101	0AA1	(2)
f	∈10101	AA1	(5)
f	10101	AA1	(3)
f	∈0101	AA1	(6)
f	0101	0A1	(3)
f	∈101	A1	(5)
f	101	1A1	(3)
f	01	A1	(6)
f	01	01	(3)
f	1	1	(5)
f	∈	∈	(6)
f	—	∈	(7)

Hence accepted

Date:

OR $(q, 0010101, \epsilon) \vdash (f, 0010101, S)$

$\vdash (f, 0010101, 0S1)$

$\vdash (f, 010101, S1)$

$\vdash (f, 010101, 0A11)$

$\vdash (f, 10101, A11)$

$\vdash (f, 10101, 1A11)$

$\vdash (f, 0101, AA1)$

$\vdash (f, 0101, 0A1)$

$\vdash (f, 101, A1)$

$\vdash (f, 101, 1A1)$

$\vdash (f, 01, A1)$

$\vdash (f, 01, 01)$

$\vdash (f, 1, 1)$

$\vdash (f, \epsilon, \epsilon)$

Q Consider the CFG

$G = (V, T, P, S)$ where

P is

$E \rightarrow T / E+T$ $\quad V = \{E, T, F\}$

$T \rightarrow F / T*F$

$F \rightarrow a / (E)$

Construct a equivalent PDA & trace out acceptance of $a+(a*a)$

Date: Page:

Design a PDA for the grammer

$G = (V, T, P, S)$ where

$V = \{S\}$

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

P consists of

$S \longrightarrow aSa$

$S \longrightarrow bSb$

$S \longrightarrow C$

check whether $w = abbcbba$ is accepted by PDA

#

P consists of

$S \longrightarrow aAS \mid bS \mid a$

$A \longrightarrow bB \mid a$

$B \longrightarrow aA \mid b$

Soln:

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

where,

$Q = \{q, f\}$

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

δ consists of :-

(1) $\delta(q, \epsilon, \epsilon) \longrightarrow (f, S)$

(2) $\delta(f, \epsilon, S) \longrightarrow (f, aAS), (f, bS), (f, a)$

(3) $\delta(f, \epsilon, A) \longrightarrow (f, bB), (f, a)$

(4) $\delta(f, \epsilon, B) \longrightarrow (f, aA), (f, b)$

(5) $\delta(f, a, a) \longrightarrow (f, \epsilon)$

(6) $\delta(f, b, b) \longrightarrow (f, \epsilon)$

(7) $\delta(f, \epsilon, \epsilon) \longrightarrow (f, \epsilon)$

$q_0 = q$ initial state.

$f = f$ final state.

$\Gamma = \{S, A, B, a, b\}$

$w = abbba$

State	Unread i/p	Stack content	Transition used
q	abbba	ϵ	—
f	abbba	S	(1)
f	abbba	aAS	(2)
f	bbba	AS	(3)
f	bbba	BBS	(3)
f	bba	BS	(6)
f	bba	bS	(4)
f	ba	S	(6)
f	ba	bS	(2)
f	a	S	(6)
f	a	a	(2)
f	ϵ	ϵ	(5)
f	—	ϵ	(7)

Date: _____ Page: _____

Pumping lemma for CFL :-

Let L be a context free language and $z \in L$. Let 'n' be the length of the string. Then, z can be decomposed into sub-strings u, v, w, x and y such that

1. $|vx| \neq \epsilon$ i.e. v, x not empty at a time.
2. $|vwx| \leq n$
3. $uv^k wx^k y \in L$, for all $k \geq 0$

v & x pumped any no. of times belongs to CFL.

Soln :-

Let $S \longrightarrow AB$

$A \longrightarrow aB \mid a$

$B \longrightarrow bA \mid b$ be the CFG

let $z = ababb$ be the string such that $z \in L$.

According to pumping lemma,

$u = a$	
$v = ba$	
$w = b$	
$x = \epsilon$	
$y = b$	

$S \longrightarrow AB$

$\longrightarrow aBB$

$\longrightarrow abABB$

$\longrightarrow ababB$

$\longrightarrow ababb$

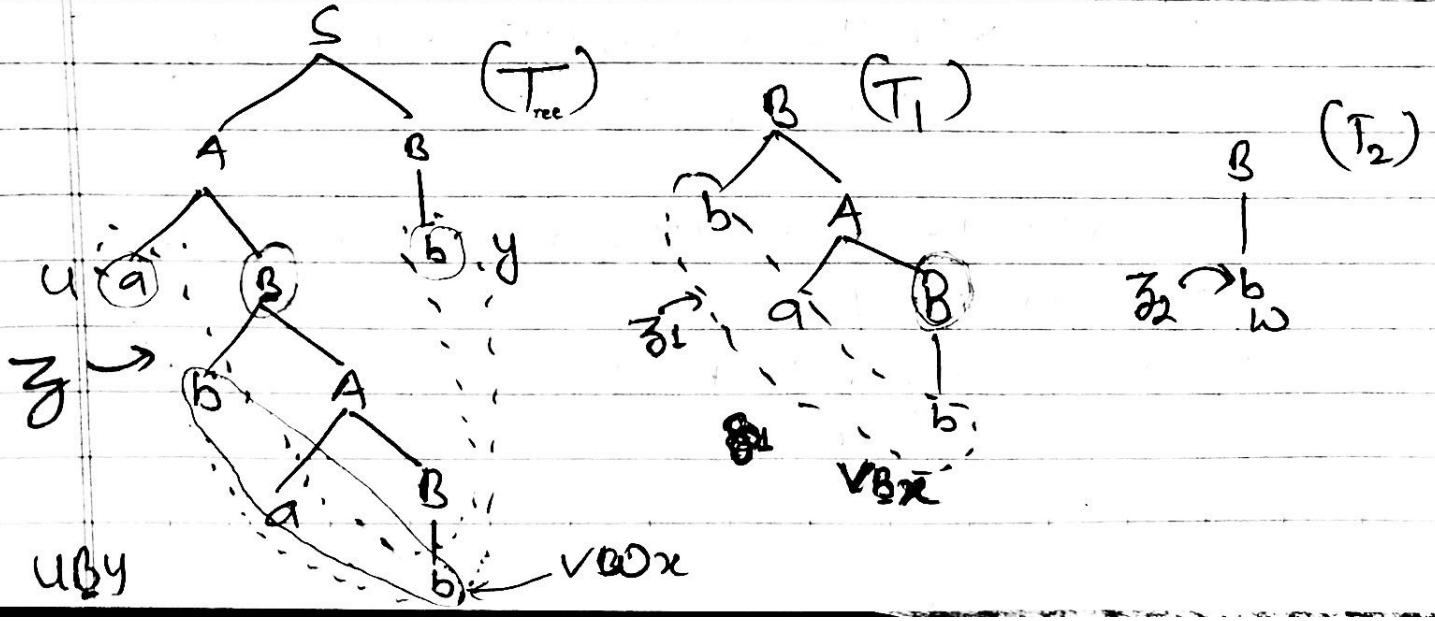
~~$A \longrightarrow AB$~~ $A \longrightarrow aB$

~~$B \longrightarrow bA$~~ $B \longrightarrow bA$

$A \longrightarrow a$

$B \longrightarrow b$

$B \longrightarrow b$



→ Here, T_1 is the proper subtree of T and T_2 is the proper subtree of T_1. Therefore, we can write.

$S \longrightarrow UB_1y$ and $B_1 = VB_2x$ and

$B_2 = W$

Since, g is the yield of S-tree and g_1 and g_2 are the yields of B-tree, we can write production rules of the form

$S \longrightarrow UBy$,

$B \longrightarrow VBx$ and

$B \longrightarrow W$

Also, ~~VBx~~ $|VWx| > |W|$ ($\because |V| \neq \epsilon$)

& ~~$|Vx| > |Vx|$~~

Using above production rules,

$S \longrightarrow UBy \longrightarrow UWy \longrightarrow UV^0Wx^0y$,

which is true for $k=0$

~~Again~~,

$S \longrightarrow UBy \longrightarrow UVBxy \longrightarrow UVWxy$,

which is true for $k=1$.

$S \longrightarrow UBy \longrightarrow UVBxy \longrightarrow UVVBxxy \longrightarrow$

UV^2Wx^2y, which is true for $k=2$

Ily, we can say that UV^kWx^ky is also true.

$\therefore UV^kWx^ky \in L$.

Date: _____ Page: _____

Q Show that the language, such that
$L = \{a^n b^n c^n : n>0\}$ is not content free.

eg:- $w = aaa\,bbb\,ccc$

such examples i.e. 3 variables cannot be dealt by CFG, or PDA. We need TM.

Solⁿ Let L be the content free language and $z \in L$.

let $z = a^p b^p c^p$

According to pumping lemma, z can be decomposed into u, v, w, x and y such that,

$u = a^q$

$v = a^r$, $r>0$

$w = a^s$

$x = a^t$, $t>0$

$y = ~~a^{p-t}~~ \cdot a^{p-(q+r+s+t)} b^p c^p$

now, for $k=2$

$uv^2wx^2y = a^q (a^r)^2 a^s (a^t)^2 a^{p-(q+r+s+t)} b^p c^p$

$= a^{p+r+t} \cdot b^p c^p$

here, $r>0$ and $t>0$, $p+r+t > p$

Therefore $a^{p+r+t} b^p c^p$ is not of the form $a^p b^p c^p$

i.e. $uv^2wx^2y \notin L$.

Hence, L is not content free ✱ proved