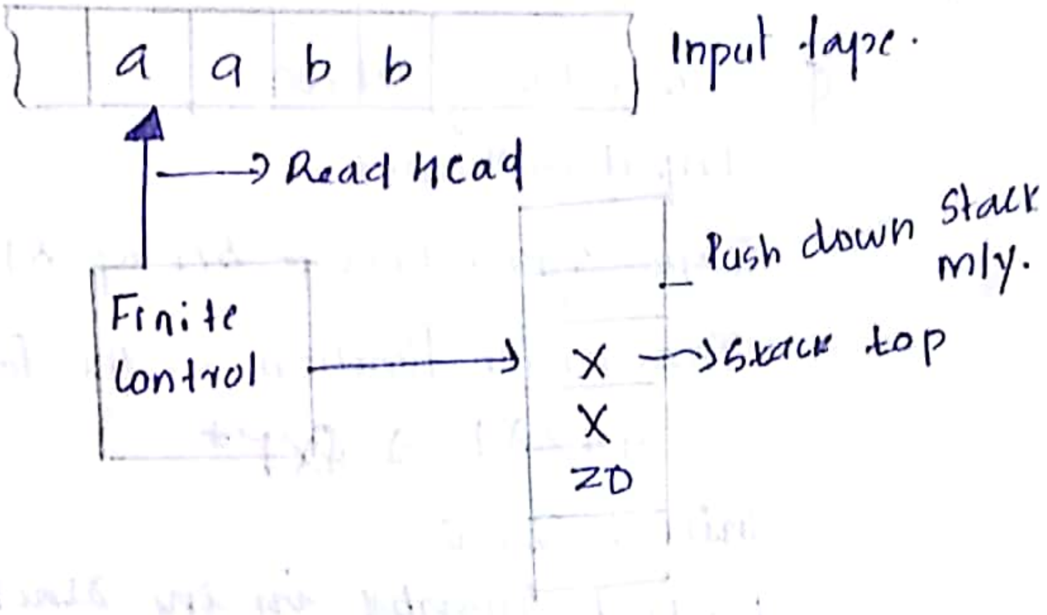
Module - 4

Pushdown Automata (PDA)



An automata accepted by CFLanguages are called PUSHDOWN AUTOMATA (PDA). The PDA is essentialy an ε-NFA with addition of a stack

$$\boxed{\varepsilon NFA + STACK = PDA}$$

A PDA accept language by two method.

1. ~~That~~ PDA accept strings by entering in an accepting state

2. PDA accept string by emptying its stack regardless of its state. it is in

3. Above fig show informal representatⁿ of a PDA.

Formal Defn

A PDA P can be formally defined as follows

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

Q = finite no: of state

Σ - Input alphabet

~~Tape sym~~ Finite set of stack symbol

Transition function in the format of ~~$Q \times \Sigma$~~

$Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$

Initial state

Initial symbol in the stack

set of final state.

PDA transition function

$$\delta(\underset{\text{current state}}{q}, \underset{\text{input symbol}}{a}, \underset{\text{stack top symbol}}{X}) = (\underset{\text{next state}}{P}, \underset{\text{stack push symbol}}{\gamma})$$

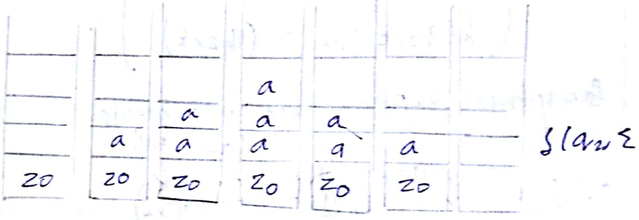
If $\gamma = \Sigma$ then stack pop operation

$\gamma = X$ then no operation.

$\gamma = YX$ then stack push Y into top of X

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

	a	a	a	b	b	b	Σ	input



Initial configurat

$\delta(q_0, a, z_0) = (q_1, az_0)$ [Push a into stack]

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

$\delta(q_1, b, a) = (q_2, \Sigma)$

$\delta(q_2, b, a) = (q_2, \Sigma)$

16/11/17 Q Construct PDA for a language $a^n b^n$ where $n \geq 1$

A) Let $P = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$ is a PDA that accept the language $L = a^n b^n$ where $n \geq 1$

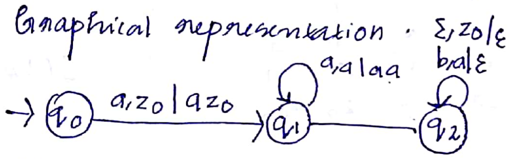
The components of P is defined as follows:

$Q = \{q_0, q_1, q_2\}$ $\Sigma = \{a, b\}$ $\Gamma = \{z_0, a\}$

$q_0 = q_0$, $Z_0 = z_0$, $F = \_\_$

$$\delta = \left\{ \begin{array}{ll} \delta(q_0, a, z_0) & = (q_1, az_0) \\ \delta(q_1, a, a) & = (q_1 aa) \\ \delta(q_1, b, a) & = (q_2, \varepsilon) \\ \delta(q_2, \varepsilon, z_0) & = (q_2, \varepsilon) \end{array} \right\}$$

Graphical representation .



(balance in Datacommunication note)

21/11/17

Equivalance of acceptance by Empty Stack and acceptanice by Final Stack

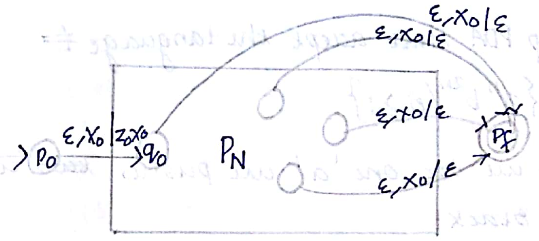
THEOREM :

If P_N be a PDA that accept language L by empty stack method then there is a PDA that accept same language L by final state method

$\boxed{L(P_N) = L(P_F)}$

PROOF

The idea behind the proof is g shown in below fig.



we use new symbol X_0 which must not be in I.
X_0 is the initial stack symbol of P_F when P_N reaches an empty stack P_F sees X_0 on the top of its stack, then it goes to final state of P_F

we also need two new states P_0 and P_F for the stimulation of PDA P_F.

The specificatn of PDA P_F is as follows.

$P_F = (Q \cup \{P_0, P_F\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, P_0, X_0 \{P_f\})$

where δ_F defined as follows.

① $\delta_F(P_0, \varepsilon, X_0) = (q_0, Z_0 X_0)$

In its starts state P_F makes spontaneous transition to the start state of P_N Pushing symbol z_0 on to the stack

② For all $q \in Q$, all the transaitn funtn present in δ_N

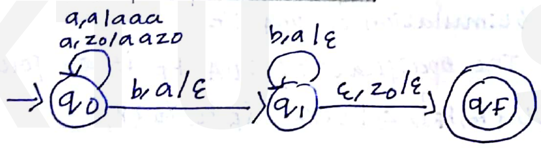
③ for all $q \in Q$, $\delta_F(q, \Sigma, X_0) = (P_F, \varepsilon)$

Q Develop PDA that accept the language ~~t~~

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

Sol: when we see one 'a' we pushes ~~two~~ 'aa' into stack.

	a	a	b	b	b	b
		a				
		a	a			
	a	a	a	a		
	a	a	a	a	a	
Zo	Zo	Zo	Zo	Zo	Zo	Zo



22/01/17 Construct PDA that accept language.

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

{	a	a	a	b	b }
			a		
		a	a	a	
Zo	Zo	Zo	Zo	Zo	Zo

$\delta(q_0, a, Z_0) = (q_1, Z_0)$

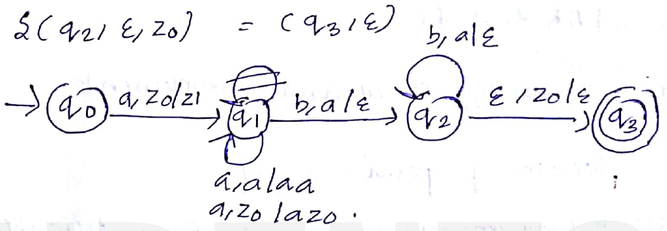
$\delta(q_1, a, Z_0) = (q_1, aZ_0)$

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

$\delta(q_1, b, a) = (q_2, a)$

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

$\delta(q_2, \varepsilon, Z_0) = (q_3, \varepsilon)$



Application of CFL

to construct parser ~~to~~ (a s/w component of compiler)

Markup languages - HTML

closure properties of CFL

P1 : CFL are closed under union.

If L_1 and L_2 are two CFL the $L_1 \cup L_2$ is also CFL

P2: CFL are closed under conatenation.

L_1 & L_2 are two CFL then $L_1 . L_2$ also CFL.

P3: CFL are closed under Kleen closure.

If L is CFL L^* is also CFL

P4: All closed CFL are closed under substitution.

P5: CFL are not closed under intersection

P6: If L is a CFL and R is a regular set, L∩R is a CFL

P7: CFL are not closed under complement

Decision properties of CFL:

~~P1~~: Testing membership in a CFL is decidable.

we can decide membership of a string w in a CFL 'L'.

Testing emptiness of CFL is decidable

we can decide/develop algorithm for whether a CFL is empty or not.

Undecidable properties of CFL.

- Is a given CFG 'G' ambigous?
- Is a given CFG G Inherently ambigious?
- Is the intersection of two CFL's empty?
- Are the two CFL the same..?