$\Rightarrow$ $a^m b^m c^m$

↳ can get generated by either L_1 or L_2, hence the CFG is ambiguous

$\Rightarrow$ In fact, the language $L = \{a^m b^m c^m\}$ is ambiguous inherently.

- The defination of CFG imposes no restriction on the right hand side of the production rule. However, complete freedom is not necessary.

Chomsky Normal Form :

A CFG $G = (V, \Sigma, P, S)$ is said to be in chomsky normal Form if every rule in P has one of the following form :

i) $A \rightarrow BC$, where A, B, C are variables and $B \neq S$, $C \neq S$.

ii) $A \rightarrow a$, where A is a variable and $a \in \Sigma$.

iii) $S \rightarrow \lambda$, where S is the start variable.

↳ iff $\lambda \in L(G)$

Theorem : Any context free language is generated by a CFG in chomsky normal form.

e.g:

$G = (V, \Sigma, P, S)$

$\Sigma = \{0, 1\}$, $V = \{A, B\}$, $S = A$

P :

$A \rightarrow BAB \mid B \mid \lambda$

$B \rightarrow 00 \mid \lambda$

Steps :

(i) Eliminate $A \rightarrow \lambda$, $A \neq S$. → Step 1

→ Even before doing this we must take care that the start variable does not come in the RHS.

→ Make a new variable as the start variable. ↳ step 0

P :

$S \rightarrow A \mid \lambda$

$A \rightarrow BAB \mid B \mid BB \mid AB \mid BA$

$B \rightarrow 00$

(iii) Eliminate rules of the form $A \rightarrow B$ where B is a variable.

$S \rightarrow BAB \mid B \mid BB \mid AB \mid BA \mid \lambda$

$A \rightarrow BAB \mid 00 \mid BB \mid AB \mid BA$

$B \rightarrow 00$

$S \rightarrow BAB \mid 00 \mid BB \mid AB \mid BA$

$A \rightarrow BAB \mid 00 \mid BB \mid AB \mid BA$

$B \rightarrow 00$

(iv) Step 4 : $A \rightarrow a_1 a_2 \ldots\ldots a_k \quad (k \geq 3)$

$S \rightarrow BA_1 \quad , \quad A_1 \rightarrow AB$

$S \rightarrow BAB$

Theorem : Any CFL can be ~~converted into~~ generated by a CFG in chomsky - normal form.

Proof : Algo to convert any CFG into chomsky - normal form.

step (i) : Introduce a new variable S_0, so that it can be made a new start variable.

step (ii) : Eliminate all $A \rightarrow \lambda$ where A is not the start variable

step (iii) : Eliminate unit rule, $A \rightarrow B$

Step (iv) : Eliminate rule, $A \rightarrow u_1 u_2 \cdots u_k$ where $k \geq 3$.

$A \longrightarrow u_1 u_2 \cdots\cdots u_k$

Replace by.

$$A \longrightarrow u_1 T_1$$
$$T_1 \longrightarrow u_2 T_2$$
$$\vdots$$
$$T_{k-2} \longrightarrow u_{k-1} u_k$$

$\Rightarrow$ After doing all these steps, we could have rules such that;

$$A \longrightarrow u_1 u_2$$

$(t_1 U_1)$ or $(U_2 t_2)$ or $(t_1 t_2)$

$T_1 \rightarrow t_1$
$A \rightarrow T_1 U_1$

$A \rightarrow U_2 T_2$
$T_2 \rightarrow t_2$

$A \rightarrow T_1 T_2$
$T_1 \rightarrow t_1, \; T_2 \rightarrow t_2$

e.g:

$G = (V, \Sigma, P, A)$, $\Sigma = \{0, 1\}$, $V = \{A, B\}$
$S = A$

P:

$A \longrightarrow BAB \mid B \mid \lambda$

$B \longrightarrow 00 \mid \lambda$

Find soln:

$S \longrightarrow \lambda \mid T_2 T_2 \mid BB \mid BA \mid AB \mid BT_1$

$A \longrightarrow BT_1 \mid T_1 T_2 \mid BB \mid BA \mid AB$

$B \longrightarrow T_2 T_2$

$T_1 \longrightarrow AB$

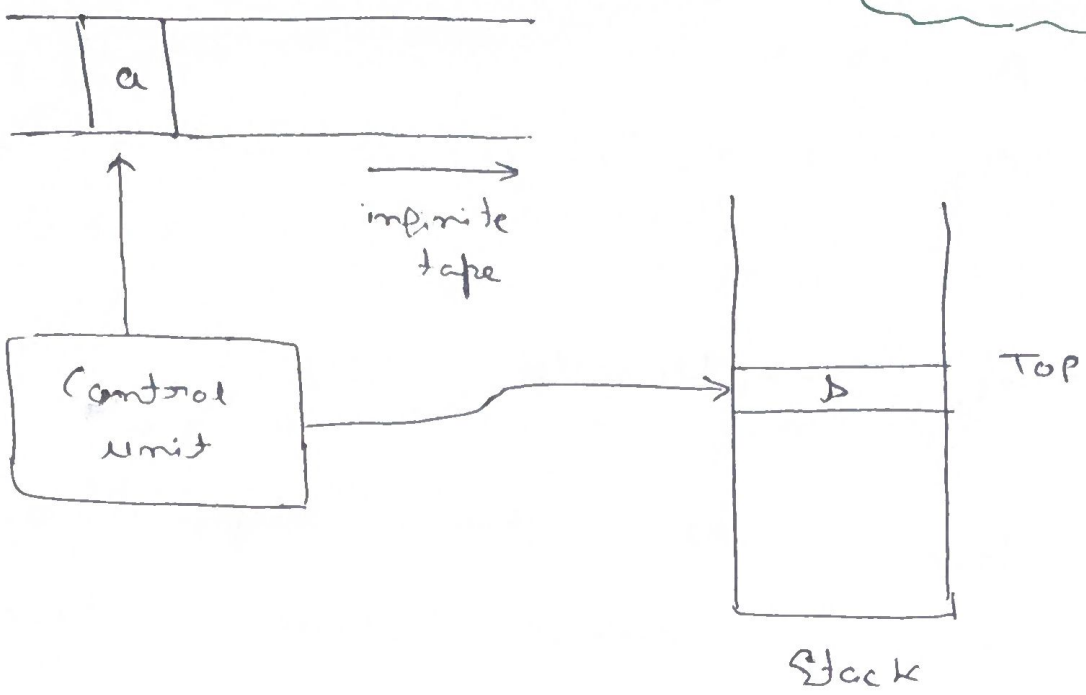
$T_2 \longrightarrow 0$

$V = \{S, A, B, T_1, T_2\}$

$\Sigma = \{0\}$.

Non-deterministic Push down Automata :

NFA + Stack



depending upon what the machine sees on the tape and the top of the stack, it takes a decision & moves to a next state.

Defn : A non-deterministic Push down automata (nPda) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where all the sets Q, Σ, Γ and F are all finite sets and

(i) Q : set of states

(ii) Σ : set of input alphabets

(iii) Γ : set of stack alphabets

(iv) δ : $Q \times \Sigma_\lambda \times \Gamma_\lambda \longrightarrow P(Q \times \Gamma_\lambda)$ is a transition function

We denote, $\Sigma_\lambda = \Sigma \cup \{\lambda\}$

$\Gamma_\lambda = \Gamma \cup \{\lambda\}$

(v) q_0 : start state (initial state), $q_0 \in Q$

(vi) F : accepting state, $F \subseteq Q$

- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is a npda & $\omega = \omega_1 \omega_2 \ldots \omega_m$ is a string & there is a sequence of states $r_0, r_1, r_2, \ldots, r_m$ and the sequence of strings $S_0, S_1, S_2, \ldots, S_m$ that we see in the stack, $\omega_i \in \Sigma_\lambda$.

s.t.

$r_0 = q_0$ [initial state]

$r_m \in F$ [accepting condn]

$\delta(\omega_i, r_{i-1}, a) = (r_i, b)$

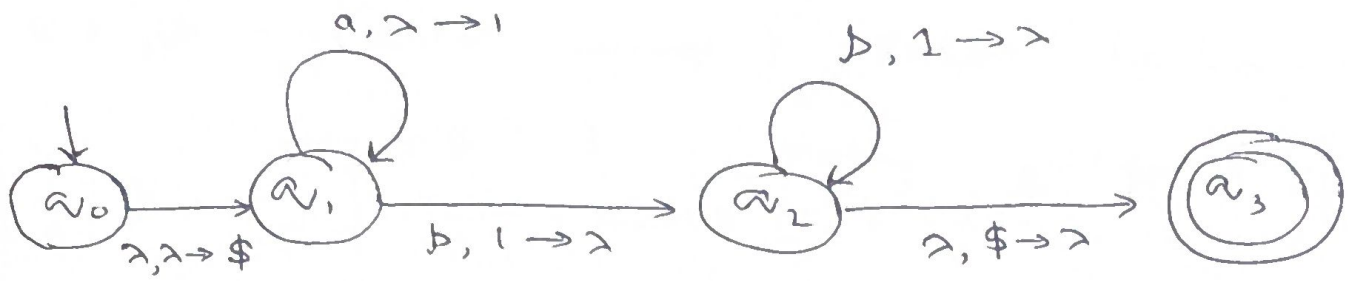
$S_i = b\underline{t}$, $S_{i-1} = a\underline{t}$

$= b \ldots\ldots$ $= a \ldots\ldots$

<u>Push</u> : $\delta(r_i, \underset{\in \Sigma_\lambda}{a}, \lambda) = (r_{i+1}, a)$

<u>Pop</u> : $\delta(r_i, \underset{\in \Sigma_\lambda}{a}, a) = (r_{i+1}, \lambda)$

e.g: $L = \{a^m b^n : m \geq 0\}$



$\delta(q_1, a, \lambda) = (q_1, 1)$

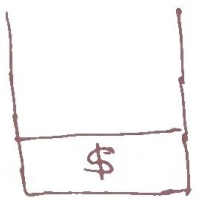
$\delta(q_1, b, 1) = (q_2, \lambda)$

$\delta(q_2, b, 1) = (q_2, \lambda)$

$\delta(q_2, \lambda, \$) = (q_3, \lambda)$

$\delta(q_0, \lambda, \lambda) = (q_1, \$)$

⇒ In the stack, before putting anything, we put a special symbol $.



Formally,

$Q = \{q_0, q_1, q_2, q_3\}$

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

$\Gamma = \{\$, 1\}$

$F = \{q_3\}$

$q_0 = q_0$

→ To make sure λ is getting accepted, $\longrightarrow$ (q_0) as well.

e.g: $L = \{ \omega \in \{a, b\}^* \mid n_a(\omega) = n_b(\omega) \}$

$L = \{ \omega \omega^R : \omega \in \{a, b\}^* \}$

$L = \{ a^i b^j c^k : \text{either } i = j \text{ or } i = k \}$

Theorem : A Language is Context Free if and only if there exists a nPDA that recognizes it.

Lemma : If a language is Context Free then there is a nPDA that recognizes it.

e.g:

$CFG = (V, \Sigma, P, S)$

$V = \{S, A, B\}$, $\Sigma = \{a, b\}$

$S \rightarrow AaB$

$A \rightarrow aA \mid a$

$B \rightarrow b$

Any string derived from this CFG is the Part of the language (CFL).

$S \rightarrow AaB \longrightarrow aaB \longrightarrow aab$

The following is an informal description of mPDA.

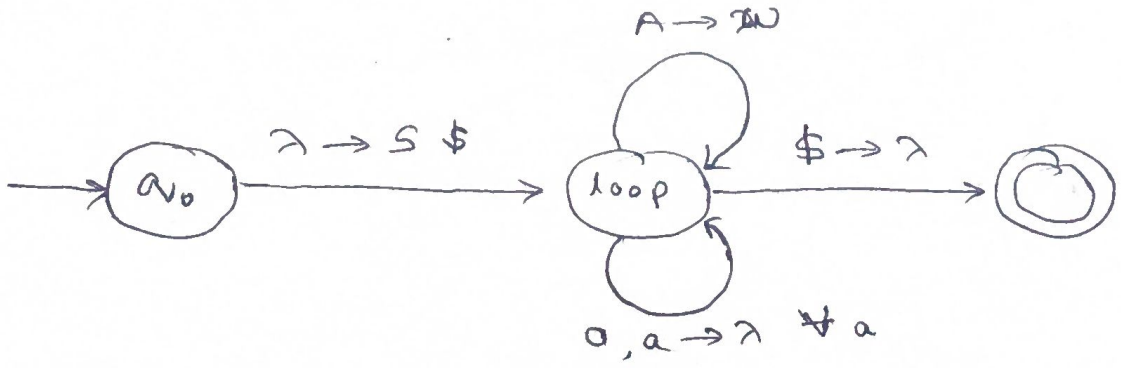
i) Place the marker symbol \$ and the start variable on the stack.

ii) Repeat the following steps forever:

→ If the top of the stack is a variable A then non-deterministically choose any production rule $A \to w$ and substitute A by w.

→ If the top of the stack is a terminal symbol a, read the next symbol from the input tape, if they match then, repeat (pop a from stack) If they do not match then reject on this branch of non-determinism.

→ If the top of the stack is \$, enter the accept state. Doing so accept the input if it has all been read.



Let's see how the mPDA looks like :

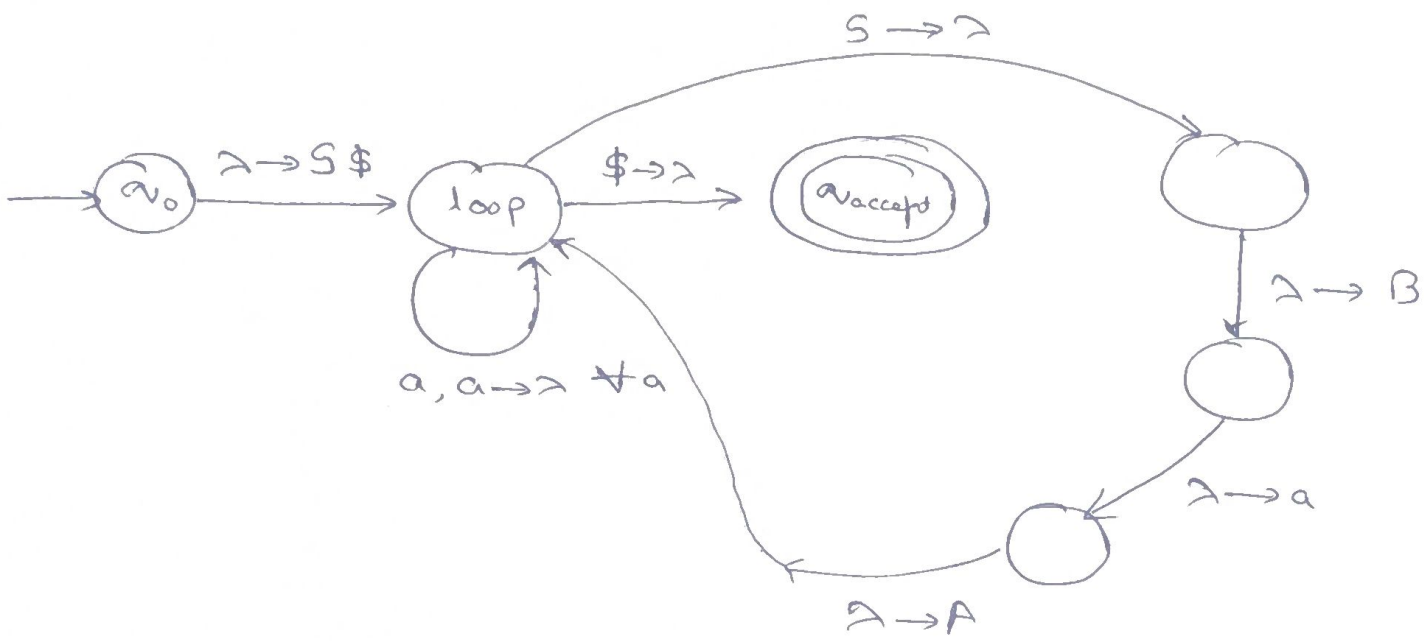
$$P = (Q, \Sigma, \Gamma, \delta, q_0, F)$$

$$G = (V, \Sigma, P, S)$$

Σ : defined , Γ : $V \cup \Sigma \cup \{\$\}$

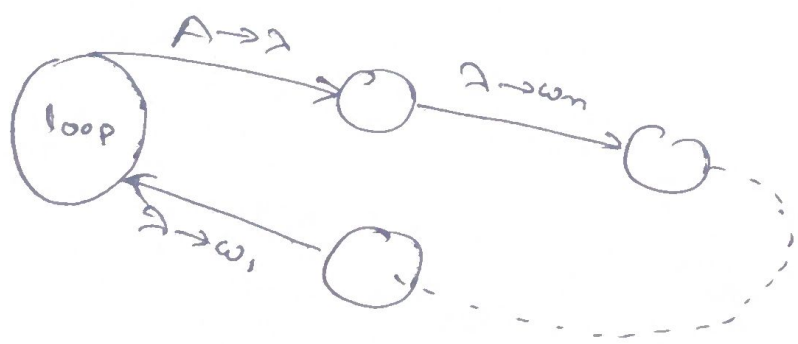
q_0 : defined F : defined

$S \rightarrow AaB$



In general :

$A \rightarrow \omega$; $\omega \rightarrow \omega_1 \omega_2 \cdots \omega_n$



$\Rightarrow$ For the other part of the proof, given a mPDA we would be requiring to give a CFG for it.