CS-303

END - SEMESTER ASSIGNMENT

NAME - TARUSI MITTAL
ROLL NO - 1901CS65

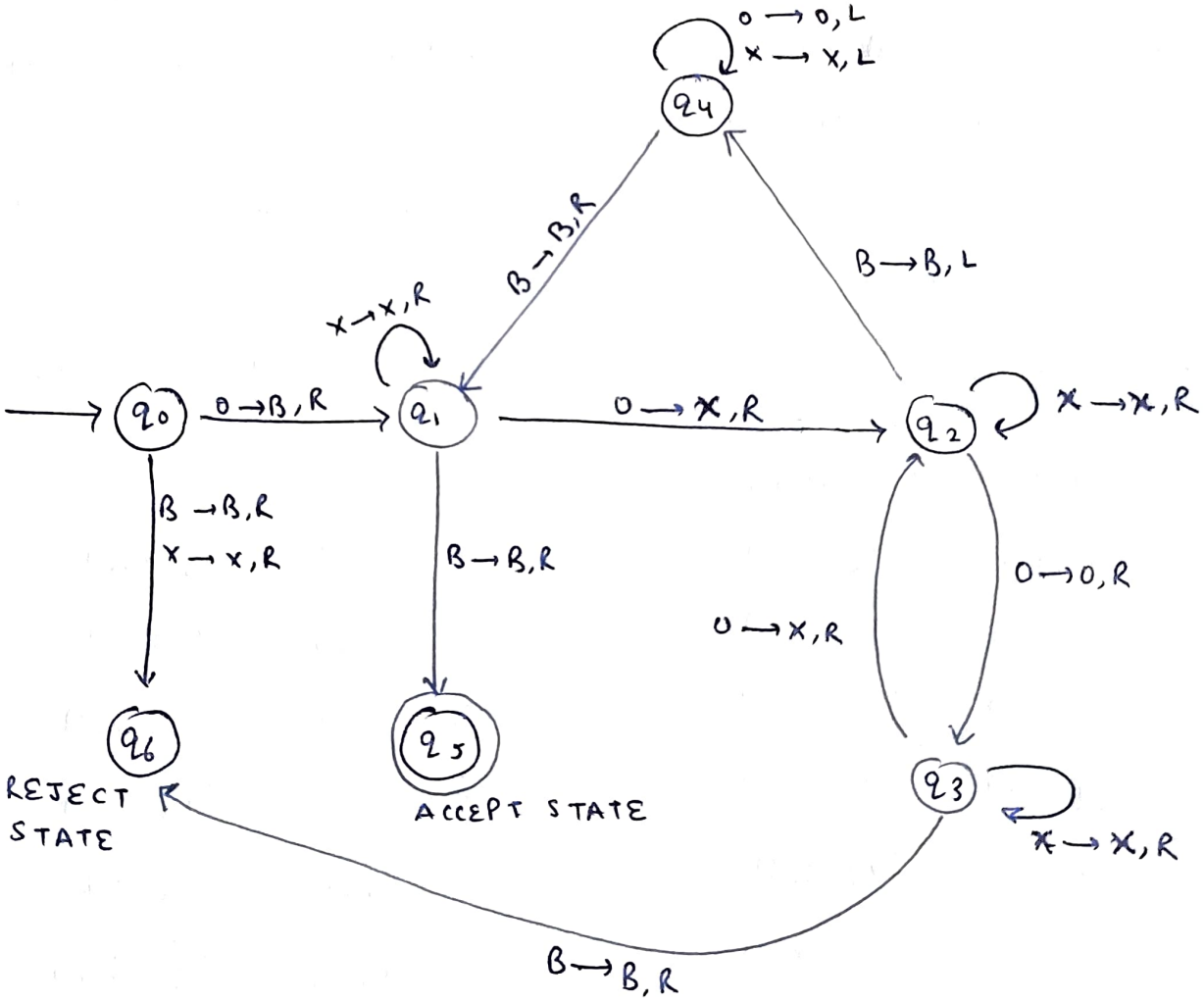
Que 1:- Construct a Turing Machine for the language.

$$L = \{0^{2^n} : n \geq 0\}$$

⇒ Let M be the turing machine which accepts the language consisting of all type of the string 0^{2^n}

$M = \{\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{0\}, \{0, X, B\}, \delta, q_0, q_5, q_6\}$

B = Blank Symbol



TURING MACHINE FOR 0^{2^n}

We know for a string of type 0^{2^n}, we can take the length to be $2k$ where $k \geq 0$. Now for 2^k length if we will keeping reducing the length by the factor of 2 then at all the stages except the last one should be of even length. The last one would be ~~even~~ odd because there only 1 will be left.

Now for the turing machine to accept a string of type 0^{2^n}; $n>0$
Let us take the string to be s.

Steps taken to create the above turing machine:

1. We will replace every alternate 0 by X, while moving from left to right.
2. If the initial state of the tape contains more than 1 odd number of 0 ie (3,5,7) then we reject it
3. If the initial state of the tape contains only 1 single odd number of 0, then we accept it.
4. Then we move from right to left and repeat the above steps.

TRANSITION FORMULA

$\delta(q_0, B) = (q_6, B, R)$
$\delta(q_0, X) = (q_6, X, R)$
$\delta(q_0, 0) = (q_1, B, R)$
$\delta(q_1, X) = (q_1, X, R)$
$\delta(q_1, B) = (q_5, B, R)$
$\delta(q_1, 0) = (q_2, X, R)$
$\delta(q_2, X) = (q_2, X, R)$
$\delta(q_2, B) = (q_4, B, L)$
$\delta(q_2, 0) = (q_3, 0, R)$
$\delta(q_3, X) = (q_3, X, R)$
$\delta(q_3, B) = (q_4, B, R)$
$\delta(q_3, 0) = (q_2, X, R)$
$\delta(q_4, X) = (q_4, X, L)$
$\delta(q_4, 0) = (q_4, 0, L)$
$\delta(q_4, B) = (q_1, B, R)$

Derivation Steps accepting the string 0000 on Tape

$\rightarrow (q_0 000B) \longrightarrow (Bq_1 000B) \longrightarrow (BXq_2 00B) \longrightarrow (BX0q_3 0B) \longrightarrow$

$(BX0Xq_2 B) \longrightarrow (BX0q_4 XB) \longrightarrow (BXq_4 0XB) \longrightarrow (Bq_4 X0XB) \longrightarrow$

$(q_4 BX0XB) \longrightarrow (Bq_1 X0XB) \longrightarrow (BXq_2 0XB) \longrightarrow (BXXq_2 XB) \longrightarrow$

$(BXXXq_2 B) \longrightarrow (BXXq_4 XB) \longrightarrow (BXq_4 XXB) \longrightarrow (Bq_4 XXXB)$

$\longrightarrow (q_4 BXXXB) \longrightarrow (Bq_1 XXXB) \longrightarrow (BXq_1 XXB) \longrightarrow (BXXq_1 XB)$

$\longrightarrow (BXXXq_1 B) \longrightarrow (BXXXBq_5)$

where q_5 is the accepting state.

$\Rightarrow$ The Turing Machine created will accept "0000".

Que 2:- R → Regular Set. ; L & L' → CFL's.

(a) $L = R$

$\Rightarrow$ Let $L_1 = L = R$ ——— (1)

With all different sets of L and R that follow condition 1. Lets suppose we are numbering the R's by the right linear grammar and R_i be the i^{th} R - Linear grammar.

Now if $L = R$ is decidable ; $L_1 = \{ (i, j) \mid L(g_i) = L(R_j) \}$ has to be recursive

But we know that the set L' ⇒

$L' = \{ i \mid L(g_i) = \Sigma^* \}$ is not recursive.

Now;

Taking some n for.

$L(R_n) = \Sigma^*$

$\Rightarrow \quad L' = \{i \mid L(g_i) = L(R_n)\} \leq m\{\langle i,j\rangle \mid L(g_i) = L(R_j)\}$

Now, as L' is not recursive $\Rightarrow$ our original set $L=R$ is also not recursive.

$\Rightarrow$ **$L = R$ is undecidable**

(b) $L = LL$

$\Rightarrow$ Given $\langle i,j\rangle$, we know the language $NVC_{ij} = valComps_{M_{ij}}$ is a CFL constructing a grammar for $NVC_{ij} \Rightarrow$

$$NVC_{ij} = \Sigma^* \iff j \notin L(M_i)$$

ie NVC_{ij} is Σ^* if $M_i(j)$ rejects or is otherwise something small.

Now,

$NVC_{ij}\ NVC_{ij} = \Sigma^*$ everytime, (not depending on $M_i(j)$.

Now but any empty string is not valid.

Therfore

$w = \epsilon w \quad \forall \ w \in NVC_{ij}$ or $w = w_1 w_2 \ \forall \ w_1, w_2 \notin NVC_{ij}$ if $w \notin NVC_{ij}$.

Considering this. $\Rightarrow$ w can be written as join of w_1 and w_2 which individually are not valid. Therfore the function maps a pair $\langle i,j\rangle$ to an index t such that

$L(G_t) = NVC_{ij} \Rightarrow$

$L_{mbr} \leq mL_g = \{i \mid L(G_i) = L(G_i)L(G_i)\}$

$\Rightarrow$ L is not recursive

$\Rightarrow$ **$L = LL$ is undecidable**

(c) $L \subseteq R$

$\Rightarrow$ We know

$L \subseteq R \iff L \cap \overline{R} = \phi$ (L is subset of R)

Now,

CFL's are closed under intersection with regular sets. Also it is decidable whether a CFG generated an empty language which means the set $L_\phi = \{i \mid L(g_i) = \phi\}$ is recursive.

Given $\langle i, j \rangle$ we can reduce L to L_ϕ by constructing a CFG (A) such that.

$L(A_t) = L(g_i) \cap L(\overline{R_j})$ and we check if t is in L_ϕ or not.

$\Rightarrow$ **$L \subseteq R$ is decidable**

(d) $L \supseteq L'$

Now, for every language L;

$L \supseteq \Sigma^* \iff L = \Sigma^*$

Using the same concept as used in 2(a), we can prove that L is not recursive which implies L is undecidable

$\Rightarrow$ **$L \supseteq L'$ is ~~not~~ undecidable**

Que 3:- Show by example that $a^l b^m c^n \mid l! = m$ or $m! = n$ is accepted by DPDA or not. $\Sigma = \{a, b, c\}$

⇒ If we see.

$L = \{a^l b^m c^n \quad l \neq m \text{ or } m \neq n\}$; we see CFL L is CFL

If L is DCFL, then

$L' = -L$ (also a DCFL

$= \{a^i b^j c^k, i, j, k \geq 0 \text{ and } i = j = k\} \cup$

$= \{w \in \{a, b, c\}^* : \text{the letters are out of order}\}$

But then;

$L'' = L' \cap a^* b^* c^*$

$= \{a^n b^n c^n, n \geq 0\}$. should also be DCFL

But certainly $a^n b^n c^n$ is not DCFL

⇒ L is context free and not deterministic context free

Therefore | L is not accepted by a DPDA |

We can also see it through like this:

If we will try to build a PDA to accept the language in the question, for now we don't know lets say that if it is deterministic or not.

So lets take the stack symbol to be ⊥ and we will take the state as q_i

where $i = 0, 1, 2, \ldots$

Taking q_0 to be the start state.

Making the transition function

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

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

Now, as soon as b arrives we have two options

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

or

$$\delta(q_0, b, a) \longrightarrow (q_2, ba)$$

Now, we need to see.

If $l \neq m \longrightarrow$ ~~Insert b~~ Pop out a

If $m \neq n \longrightarrow$ keep b and later when c comes we will pop it out.

⇒ That this step is a non deterministic step.

⇒ We cannot construct a DPDA for it.

Hence the Language given in the question cannot be accepted by a DPDA.

Que 4:-

$L \rightarrow$ Recursively enumerable

$\bar{L} \rightarrow$ Non Recursively enumerable.

$L' = \{0w \mid w \text{ is in } L\} \cup \{1w \mid w \text{ is not in } L\}$

Let us suppose, L' to be recursively enumerable.

Then we could get a turing machine M which accepts L'.

Now, we could design a TM M for $\bar{L}$ given input w. as:

Given input w, M changes its input to $1w$ and simulates the hypothetical TM for L'. If that TM accepts then w is in $\bar{L}$, so M should also accept w.

If it rejects then, neither will M. $\Rightarrow$

Thus M would accept exactly $\bar{L}$, which contradicts the fact that $\bar{L}$ is non-Recursively enumerable.

Therefore $\Rightarrow$ our assumption of $L' \rightarrow RE$ is false

$\Rightarrow$ **L' is Non Recursively Enumerable**

Que 6:-

(a) $a^n b^m \mid 2^n = 3m$; $n \geq 0$ and $m > 0$ $\quad \Sigma = \{a, b\}$

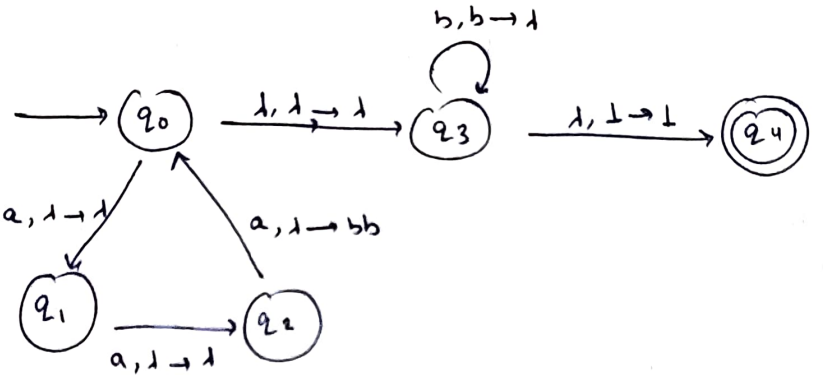
Let $6k = 2n = 3m$ ($n = 2n = 3m$, $\Rightarrow$ n is divisible by 2 & 3)
it must be divisible by 6

$\Rightarrow n = 3k \quad m = 2k.$

$L = \{a^{3k} b^{2k} \mid k \geq 0\}$

Taking the stack start symbol to be $\perp$

PDA.



$M = \{\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \{b, \perp\}, \delta, q_0, \perp, q_4\}$.

Logic :→ If string is empty it will keep on moving and will reach q_4 (final state)

Otherwise:

1. It will check for a. Whenever ~~the~~ we find 3 a's continuously it will push 3 a's into the stack.

2. After that for every b it will encounter, it will remove one, ~~one~~ b from the stack.

3. If at the end only $\perp$ (stack start symbol) is present, it means it has accepted that string.

The Transition function

$\delta(q_0, a, \lambda) = (q_1, \lambda)$ → for 1st a it will look for another a

$\delta(q_1, a, \lambda) = (q_2, \lambda)$ → for 2nd a it will see for 3rd.

$\delta(q_2, a, \lambda) = (q_0, bb)$ → for 3 a's that it will see it pushes 'bb' to the stack. and continues the process till a's are finished

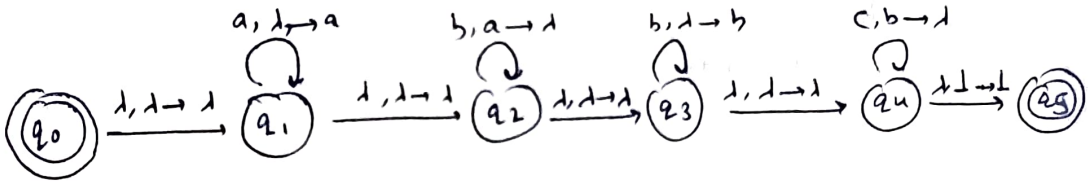
$\delta(q_0, \lambda, \lambda) = (q_3, \lambda)$ → it simply is transition function

$\delta(q_3, b, b) = (q_3, \lambda)$ → for any b that we reach it pop one b out of the stack

$\delta(q_3, \lambda, \perp) = (q_4, \perp)$ → while reaching the end of string if the stack has only $\perp$ left and we reached q_4 state it implies that, that string has been accepted.

(b) $L = a^i b^j c^k \mid i, j, k \geq 0$ and $i+k = j$ $\quad \Sigma = \{a, b, c\}$

⇒



We can write the string $a^i b^{i+k} c^k = a^i b^i b^k c^k$.

$M = \{\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c\}, \{a, b, c, \perp\}, \delta, q_0, \perp, \{q_0, q_5\}\}$.

Logic :- If string is empty it will get accepted at q_0, in other cases it moves to q_1 via λ- transition.

Now;

For i times our PDA will push a to the stack corresponding to every a in the string at stage q_1

If string has no a's or all a are done processing we will move to state q_2 via λ- transition.

Now, at q_2, as many b's will be there, that many times we pop a out of stack until the start element of the stack becomes ⊥, then via λ- transition we move to q_3.

Now, at q_3, for remaining no's of b we will push them in the stack.
If 0 b are left or all b are done processing we will go to q_4 via λ- transition.

At q_4, for every c encountered in the string we will pop b out of stack.
Now if we end the string and we reach ⊥ in stack (start symbol) we would go to q_5 which is our accepting state.

The Transition Function

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

$\delta(q_1, a, \perp) = (q_1, a)$ → for every a, put 1 a in stack

$\delta(q_1, \perp, \perp) = (q_2, \perp)$

$\delta(q_2, b, a) = (q_2, \lambda)$ → for every b, pop a

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

$\delta(q_3, b, \perp) = (q_3, b)$ → for remaing b, push b in stack.

$\delta(q_3, \lambda, \perp) = (q_4, \perp)$

$\delta(q_4, c, b) = (q_4, \lambda)$ → for every c, pop b out of stack

$\delta(q_4, \lambda, \perp) = (q_5, \perp)$ → Accepting state