24/11/2021

CS303 End-Sem TANISHQ MALU 1901CS63

Q1

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

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

⇒ Suppose we have a string of length $2^k : k \geq 0$.
If we keep dividing it in halves then length would become $2^k \to 2^{k-1} \to 2^{k-2} \to \ldots\ldots 2^1 \to 1$

At all stages the length of string would be even except the last stage where it is odd and its length 1.

Thus, we can use the above fact to create a turing machine to accept $0^{2^n}; n \geq 0$

Formally

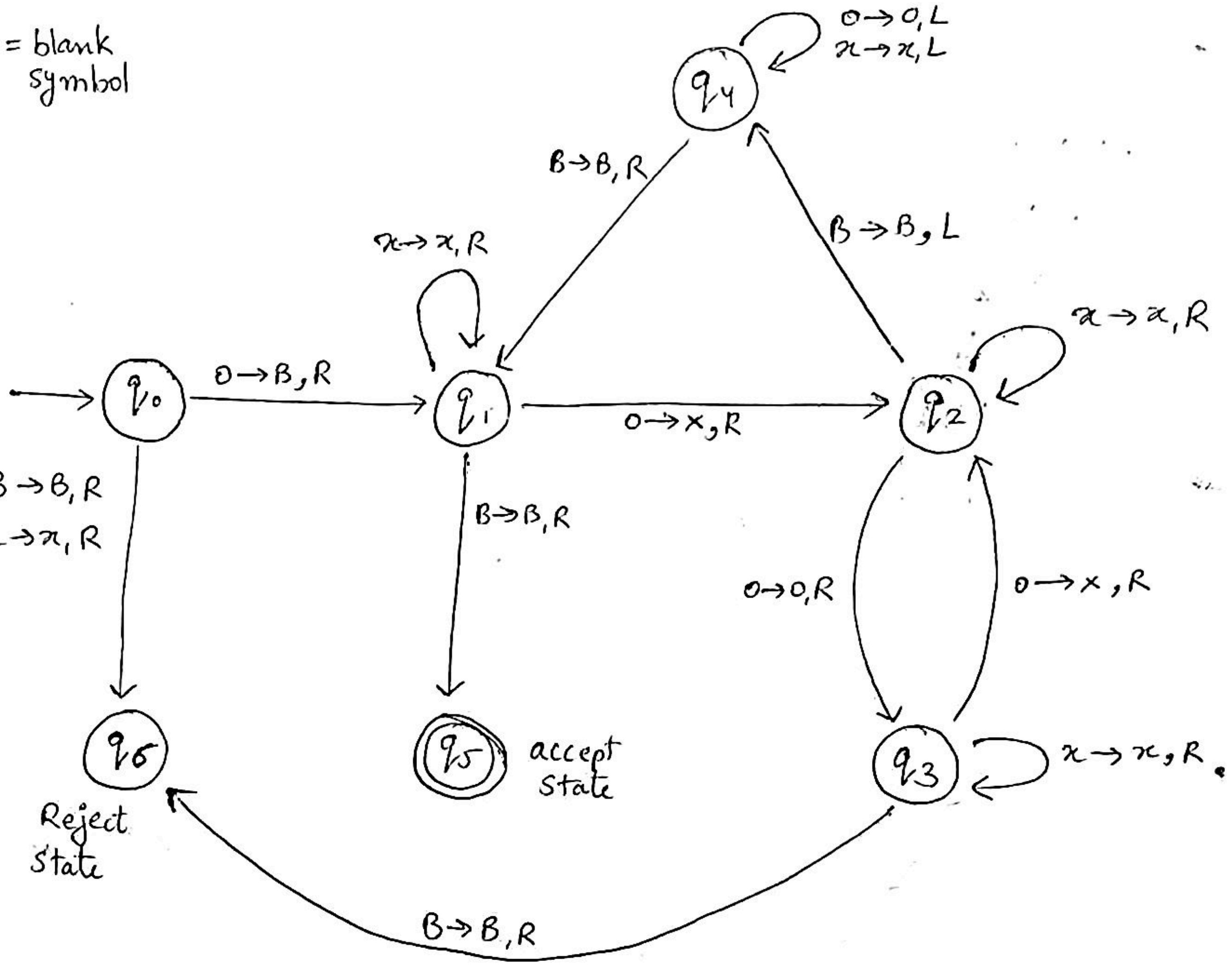
M ⇒ on the input string s:

1. Move from left to right, replacing every alternate 0 by x.
2. If in initial stage the tape contains single "0", then move to accept state
3. If in inital stage the tape contains more than single odd number of "0" then reject.
4. Move back to left end of tape
5. Recurse.

Thus, it will, in each iteration, replace half number of 0 by x and keep a track of no. of 0's seen. If its odd and greater than 1 simply reject.

Q1 $M = \{\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{0\}, \{0, x, B\}, \delta, q_0, q_5, q_6\}$

B = blank symbol



TM for 0^{2^n}

Transition function

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

Lets now input 0000 to our turing machine:-

$\rightarrow q_0 0000$

$\rightarrow B q_1 000$

$\rightarrow B \times q_2 00.$

$\rightarrow B \times 0 q_3 0$

$\rightarrow B \times 0 \times q_2 B$

$\rightarrow B \times 0 q_4 \times B$

$\rightarrow B \times q_4 0 \times B$

$\rightarrow B q_4 \times 0 \times B$

$\rightarrow q_4 B \times 0 \times B$

$\rightarrow B q_1 \times 0 \times B$

$\rightarrow B \times q_1 0 \times B$

$\rightarrow B \times \times q_2 \times B$

$\rightarrow B \times \times \times q_2 B$

$\rightarrow B \times \times q_4 \times B$

$\rightarrow B \times q_4 \times \times B$

$\rightarrow B q_4 \times \times \times B$

$\rightarrow q_4 B \times \times \times B$

$\rightarrow B q_1 \times \times \times B$

$\rightarrow B \times q_1 \times \times B$

$\rightarrow B \times \times q_1 \times B$

$\rightarrow B \times \times \times q_1 B$

$\rightarrow B \times \times \times B q_5$

Reached accept state

Thus the above TM accepts 0000.

Q2

L and L^g are CFL (given)

R (regular expression or right linear grammar)

1.) $L = R$

let $L_1 = \quad L = R \quad -(1)$.

Thus there exists different L_s and R_s such they follow condition (1). let us assume, we are numbering the $R's$ and let R_i be the i^{th} right linear grammar $\therefore$

$$L_1 = \{ (i, j) \mid L(g_i) = L(R_j) \}$$

For this we know that the set $L^g \Rightarrow$

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

Now,

Choose some k such that

$$L(R_k) = \Sigma^*$$

$\therefore L^g = \{ i \mid L(g_i) = L(R_k) \} \leq_m \{ (i,j) \mid L(g_i) = L(R_j) \}$

$\therefore$ Since L^g is not recursive, our original set $L=R$ is not recursive.

Hence it is not decidable.

Q2 b

$L = LL$

Consider the language

NVC_{ij} = Val Comps M_{ij}

we know that it is a CFL and given $\{i, j\}$ we can effectively construct a grammar for NVC_{ij}. Observe that

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

i.e NVC_{ij} is Σ^* if $M_i(j)$ rejects and is something smaller otherwise. Now lets suppose

$NVC_{ij}\ NVC_{ij} = \Sigma^*$ in all cases, whether $M_i(j)$ accepts or rejects. To visualise, note that empty string is not valid. Thus any w can be written

$$w = \epsilon w \quad \forall \quad w \in NVC_{ij}$$

or

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

In 2nd case, w can always be expressed as concatenation of w_1 and w_2 neither of which itself is valid. Thus the function maps a pair $\langle i, j \rangle$ to an index k such that

$L(G_k) = NVC_{ij}$ yielding

$L_{mbr} \leq_m L_g = \{ i \mid L(G_i) = L(G_i)\, L(G_i) \}$

Proving L is not recursive

Thus It is not <u>decidable</u>.

c) $L \subseteq R$

we know that since L is subset of R

$$L \subseteq R \iff L \cap \bar{R} = \phi$$

we know that CFL are closed under intersection with regular sets. Also it is decidable whether a CFG generates an empty language i.e the set

$$L_\phi = \{ i \mid L(g_i) = \phi \}$$ is recursive.

our language $L_1 = L \subseteq R$ can be easily reduced to L_ϕ. Given $\langle i, j \rangle$ we can construct a ~~CF~~ CFG, x such that

$$L(x_t) = L(g_i) \cap \overline{L(R_j)}$$

and then simply check if t is in L_ϕ or not.

Thus $L \subseteq R$ is <u>decidable</u>.

d) $L \supseteq L'$

we know that for any L

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

using the same logical inferences from (Qa) we can show that it is not recursive. Hence it is <u>not decidable</u> as well

Q3 $L = \{a^l b^m c^n \mid l \neq m \text{ or } m \neq n\}$

We have to check wether it is accepted by DPDA or not. Lets try to build a PDA to accept above language. As of now we don't know if its ~~ded~~ deterministic or not. Let z be the stack symbol. and q_i be the states for $i = 0, 1, 2, \ldots$ Let q_0 be the start state.

Transition function

1. $\delta(q_0, a, z) = (q_0, az)$
2. $\delta(q_0, a, a) = (q_0, aa)$ (input all "as" till first b arrives)
3. Now when a "b" arrives we have 2 possibility

$(q_0, b, a) \rightarrow (q_1; \epsilon)$.

$\searrow (q_2; ba)$

If we have to check $l \neq m$ we need to pop "a" for "b" and if we have to check $m \neq n$ we need to keep b in stack so that we can pop it for "c".

Thus it is a non-deterministic step and hence a DPDA can't be constructed for it. Further :-

4.) $(q_1, b, a) \rightarrow (q_1, \epsilon)$

5.) $(q_1, c, a) \rightarrow (q_f, \epsilon)$

6.) $(q_1, b, z) \rightarrow (q_f, \epsilon)$

7 $(q_2, b, b) \rightarrow (q_2, bb)$

8 $(q_2, c, b) \rightarrow (q_3, \epsilon)$

9 $(q_3, c, b) \rightarrow (q_3, \epsilon)$

10) $(q_3, c, a) \rightarrow (q_f, \epsilon)$

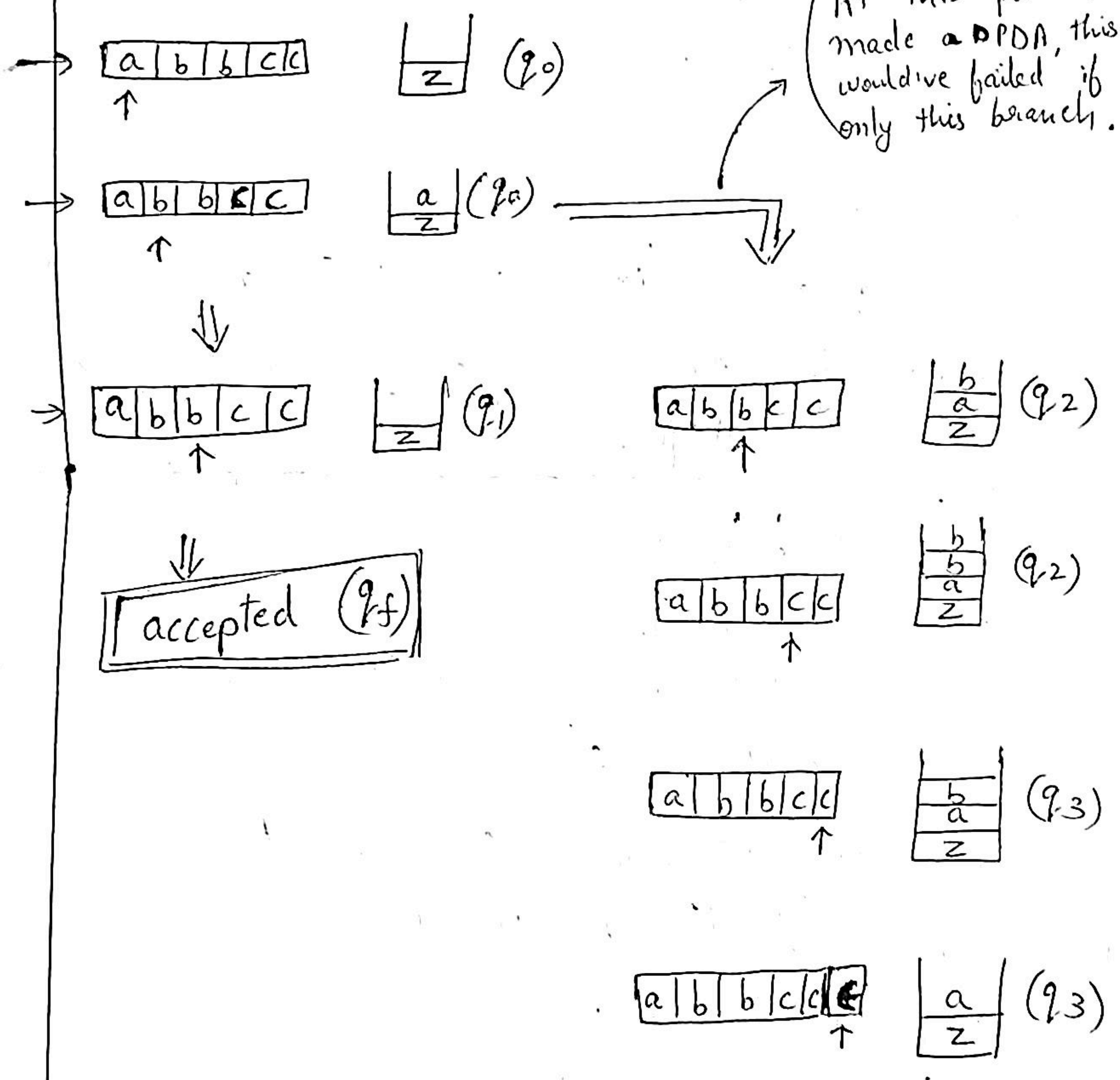
11) $(q_3, \epsilon, b) \rightarrow (q_f, \epsilon)$

It is clear that this language can't be accepted by a DPDA. Let us go through an example :-

Let $s = abbcc$ ∴ $l=1$ $m=2$ $n=2$.

here $l \neq m$ thus it should be accepted by by our NPDA.

(At this point, if we have made a DPDA, this string would've failed if DPDA had only this branch.)



No valid transition left hence in reject state

More formally:-

$L = \{a^l b^m c^n \quad l \neq m \text{ or } m \neq n\}$ given L is CFL.

If L is DCFL then

$L' = \neg L$ is also DCFL

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

$\{w \in \{a, b, c\}^* :$ in all random order$\}$

Similarly

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

$= \{a^n b^n c^n, \; n \geq 0\}$.

is also DCFL

But we know $a^n b^n c^n$ is not DCFL.

$\therefore$ L is context free and not DCFL

Thus L is not accepted by a DPDA.

Q4 given: L is recursively enumerable
$\bar{L}$ is non-recursively enumerable

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

Let us assume that L' is recursively enumerable.
Let there exsist a turing Machine M_1 which
accepts L'. Given the input w.
we could design TM M for $\bar{L}$ as follows.
M changes its input to 1w and simulated
TM of L' i.e M_1. If M_1 accepts, then
w is in $\bar{L}$ so M should also accept w.
If M_1 rejects then so does M. Thus, M would
accept. exactly what is in $\bar{L}$ which contradicts
the given fact that $\bar{L}$ is non-RE.

Thus our assumption that L' is RE is false. Hence
L' is non-recursively enumerable.

Q6 $a^n b^m \mid 2n = 3m$; $n \geq 0$ and $m \geq 0$

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

given: $2n = 3m$

$\therefore \frac{n}{3} = \frac{m}{2} = k$ (let) $k \geq 0$

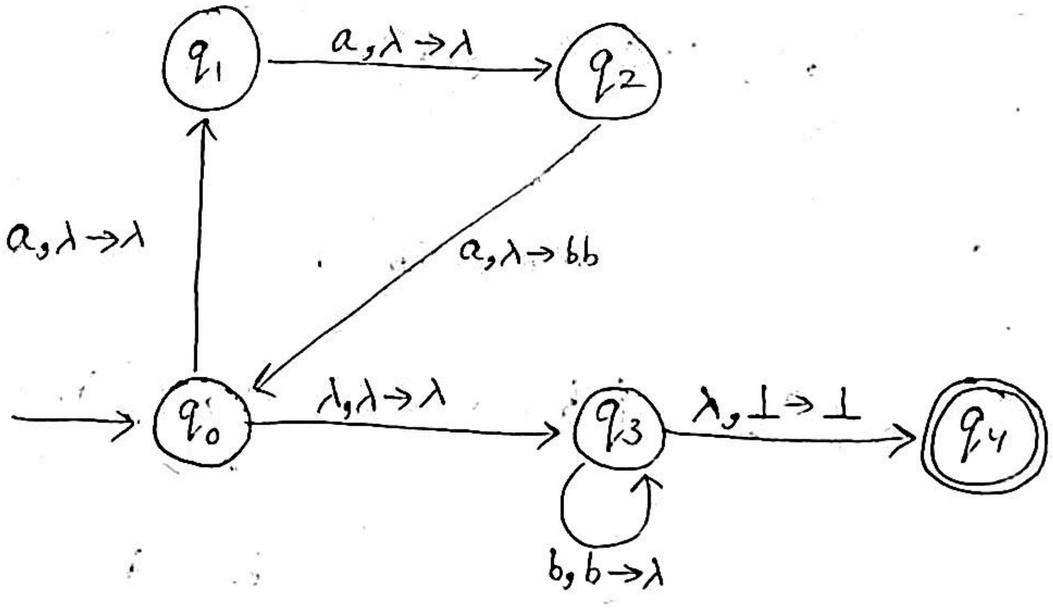
$\therefore n = 3k$

$m = 2k$

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

Stack start symbol = $\perp$

Required PDA



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

The basic idea behind PDA is to count number of a's (modulo 3). Each time it sees three a's, it pushes two b's onto stack. Nondeterministically the PDA can go from q_0 to q_3 and then for each "b" it removes one "b" from stack.

Thus in the end if stack is empty (only stack start symbol left) then the string will get accepted.

<u>Transition function</u>

$\delta(q_0, a, \lambda) = (q_1, \lambda)$
$\delta(q_1, a, \lambda) = (q_2, \lambda)$
$\delta(q_2, a, \lambda) = (q_0, bb)$

For every 3 a's encountered it will push two b's in stack

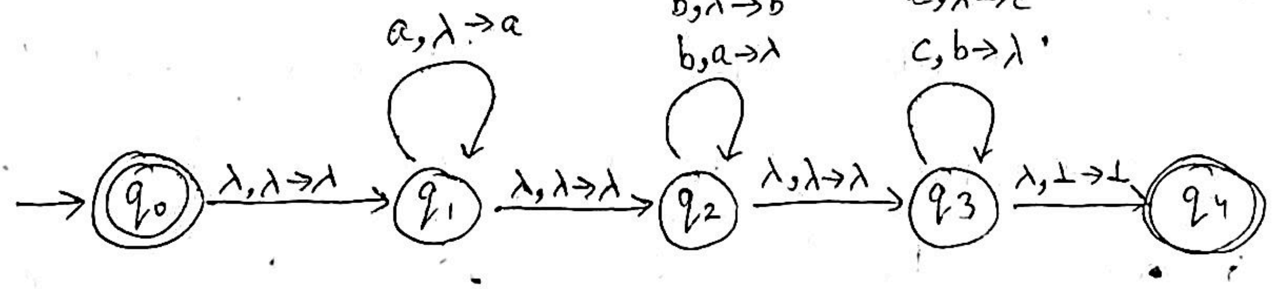
$\delta(q_0, \lambda, \lambda) = (q_3, \lambda)$

$\delta(q_3, b, b) = (q_3, \lambda)$ } it will pop one "b" whenever a "b" is encountered

$\delta(q_3, \lambda, \perp) = (q_4, \perp)$ } If we reach end of string and stack has only "$\perp$" then it goes to accept state.

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

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



given a string $a^i b^j c^k$, it can be represented as

$a^i b^i b^k c^k$ since $j = i + k$.

now if $i = k = 0$ then our string will be empty and can get accepted at initial state itself. If i or $k > 0$ then we can do a λ- transition from q_0 to q_1

<u>At q_1,</u>

If $i > 0$ then our PDA will simply push the incoming "as" into the stack. via $\delta(a, \lambda \to a)$.

If $i = 0$ or all a's have been encountered we can go to q_2 by a λ- transition

<u>At q_2,</u>

If stack consists of "as" and "b" is encountered in string, pop "a's" until we reach bottom of stack to make sure "$\underline{a^i b^i}$" has been achieved.

After this, if more "b's" are encountered simply push in the stack. Then go to q_3 by λ transition

<u>At q_3,</u>

If stack consists of "bs" and "c" is encountered in string, pop "bs" till we reach bottom of stack. If no more characters are left simply go to accept state by $(\lambda, \perp \to \perp)$ else string gets rejected for any other case.

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

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

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

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

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

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

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

$$\delta(q_3, c, b) = (q_3, \lambda)$$

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

———×———×———