21-04-2021 CSPC41 - Automata 106119100

Rajneesh Pandey

Cycle Test - 2

Question (1)

(a) <stmt> → <while - stmt>

→ while <bool-expr> do <stmt>.

→ while <arith-exp> <compare-op> <arit-exp> do <stmt>

→ while <var> <compare-op> <arit-exp> do <stmt>

→ while x <compare-op> <arith-exp> do <stmt>

→ while x ≤ <var> do <stmt>

→ while x ≤ y do <stmt>.

→ while x ≤ y do <begin-stamt>

→ while x ≤ y do begin <ass-gn-stmt>; <stm-last> end

→ while x ≤ y do begin x := (<arit-emp> <arith-op> <arith-exp> <stmt-list> end.

→ while x ≤ y do begin x := (x+1), <stm> end.

→ while x ≤ y do begin x := (x+1), y := (<var> <arith-op> <arith-exp> end

→ $\boxed{\text{while } x \leq y \text{ do begin : } (x+1),\ y := (y-1)}$

hence the required expression derived.

Question ② :

Given grammar : G's production rules.

$S \rightarrow AC \mid B$

$A \rightarrow a$

$B \rightarrow AB \mid BC$

$C \rightarrow CA \mid BC \mid \epsilon$

$E \rightarrow aA \mid \epsilon$.

→ To simplify we need to remove

(i) null transition

(ii) remove unit product

(iii) production that are unreachable

so,

Non terminal E is unreachable remaining them,

so,

$S \rightarrow AC \mid B$

$A \rightarrow a$

$B \rightarrow AB \mid BC$

$C \rightarrow CA \mid BC \mid \epsilon$.

(ii) remaining the null production rules $G \to G$. Grammer is

$S \to AC \mid B \mid A$

$A \to a$

$B \to AB \mid BC \mid B$

$C \to CA \mid BC \mid A \mid B$

(iii) Remove production rules of non-terminal B, as it doesnot reach the final state.

$S \to AC \mid A$

$A \to a$

$C \to CA \mid A$

(iv) now, substitute $A \to a$,

$S \to ac \mid a$

$C \to ca \mid a$

Then simplified Grammer G

$S \to ac \mid a$

$C \to ca \mid a$

Language generated by this grammer

$L = \{ \omega : \Sigma = \{a\} : \omega = a^n \ n \geq 1 \}$.

The regular expression is of the form language is $\boxed{L = a^+}$

Question - 3

Step 1 for all balanced Parenthesis.

Apply construction of Lemma, to get rid of ϵ - and unit production.

$$\boxed{S \rightarrow [S] \mid SS \mid [\,]}$$

Step 2 adding new nonterminals A, B & replace

$$\boxed{S \rightarrow ASB \mid SS \mid AB, \quad A \rightarrow [, \quad B \rightarrow].}$$

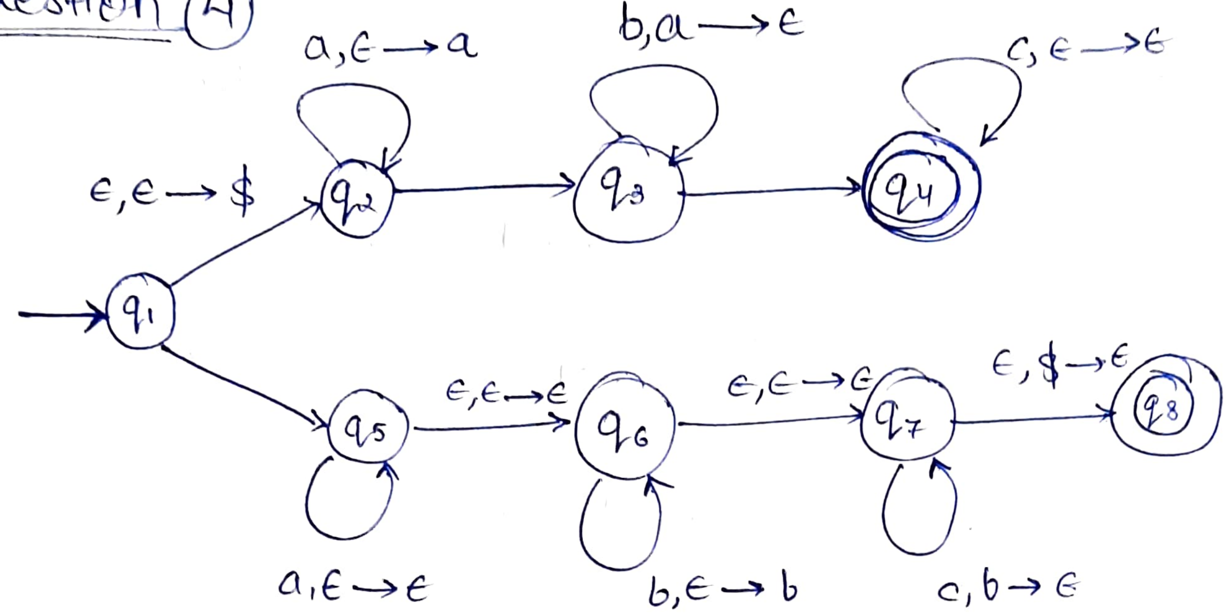
Step-3

Add a new nonterminal C & replace S→ASB

$$\boxed{S \rightarrow AC \quad \text{and} \quad C \rightarrow SB.}$$

this is the CNF grammer for the set of non-null string.

Question (4)



The above PDA has a non-deterministic branch at q_1.

If the string $a^i b^j c^k$ with $i = j$, then PDA $\Rightarrow q_1 \rightarrow q_2$

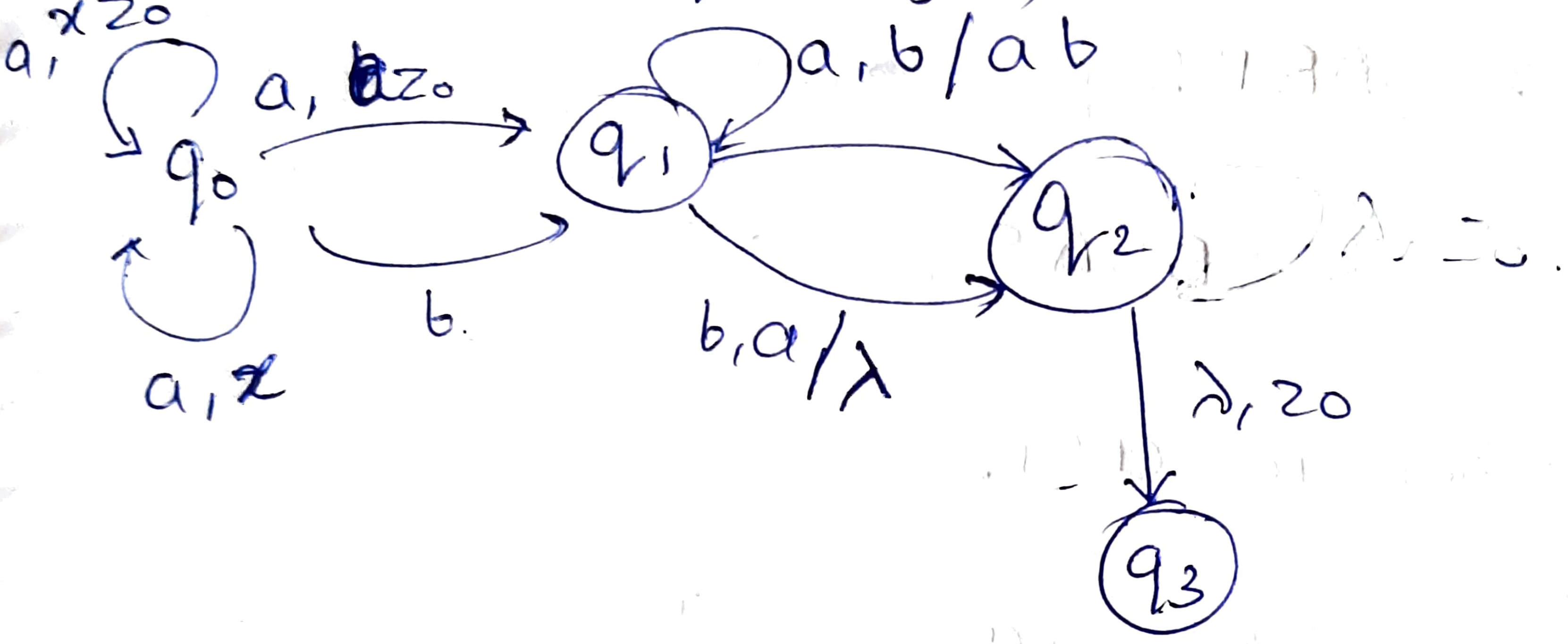
if string $a^i b^j c^k$ with $j = k$, then PDA $\Rightarrow q_1 \rightarrow q_5$

PDA : $(Q, \Sigma, \Gamma, \delta, q_1, F)$

- $Q = \{q_1, q_2, \dots q_8\}$
- $\Sigma = \{a, b, c\}$
- $\Gamma = \{a, b, \$\}$ (use \$ to mark bottom of stack)
- transition function $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow P(Q \times \Gamma_\varepsilon)$
- q_1 starting state.
- $F = \{q_4, q_8\}$.

Question 5

(a subset of $\{a,b\}^*$) accepting state q



Question ⑥

$L = \{w$ belongs to $\{a, b\}^* : n_a(w) = n_b(w)$; w doesnot contain substring bab$\}$

Step ①: here we have to match $n_a(w)$ and $n_b(w)$

So, we have to find context free grammer, so that when "a" comes "b" will also come.

also, emfy string accepted.

$$\boxed{S \rightarrow \epsilon} \longrightarrow ①$$

now,

add "a" or "b"

So, $\boxed{S \rightarrow SASBS}$ → (2)

and $\boxed{S \rightarrow SBSAS}$ → (3)

are production in CFG.

where, $\boxed{A \rightarrow a}$ — (4)

$\boxed{B \rightarrow b}$ — (5)

Step - (2):

CFG of language L will be

$\boxed{G = (N, T, P, S)}$

where

N: {S, A, B}, T = {a, b}, P = $\begin{bmatrix} S \rightarrow \epsilon \mid SASBS \mid SBSAS \\ A \rightarrow a \\ B \rightarrow b \end{bmatrix}$

Step - (3)

using induction shows, "S" derives the word w of language L.

(i) If length of w is less than 2.

$\boxed{|w| < 2}$

There are no words of length 1 in the language $n_a(w) = n_b(w)$, so only ϵ is accepted by L.

(ii) also, $\boxed{w = bab}$ not accept by L

because 2 b's and 1 a are not equal

(iii) Now $w = bu$, such that "w" have more "a" than "b"

so, $u = s_1 a s_2$.

both s_1 and s_2 are strictly shorter than w.

which implies that "w" has always equal no. of a's and b's.

Hence, the language of CFG is accepted by given language

there fore "L" is context free.