FORMAL LANGUAGE AUTOMATA

CT-2

Name: Rahul Goel
Reg No: RA1911030010094
Section: 02.

① Q17

(i)

PDA

* In PDA there may exists more than one transition for each input symbol.
* Table may contains multiple defined entries
* There is no epsilon transition, meaning that you're not allowed to change states without consuming anything from the input.

NPDA

* In NPDA there may exists exactly one transition for each input symbol.
* Table contains single entries.
* There is epsilon transition.

(iii) w.r.t to the above statements, language formed,

$$L = \{ a^{(2*m)} c^{(4*n)} d^n b^m \mid m,n$$

Eg:

Input : aa cccc db

Output : Accepted.

Input : aa c c c db

Output : Not Accepted

Case 1: m = 0 - In this cases the input string will be of the form $\{ c^{(4*n)} d^n \}$. In this condition, keep on pushing c's in the stack until we encounted with d. keep on popping cccc's until all the c's of the string are processed. If we reach to the end of the input string and the stack becomes empty then reached to the final state.

Case-2 m=0 :-

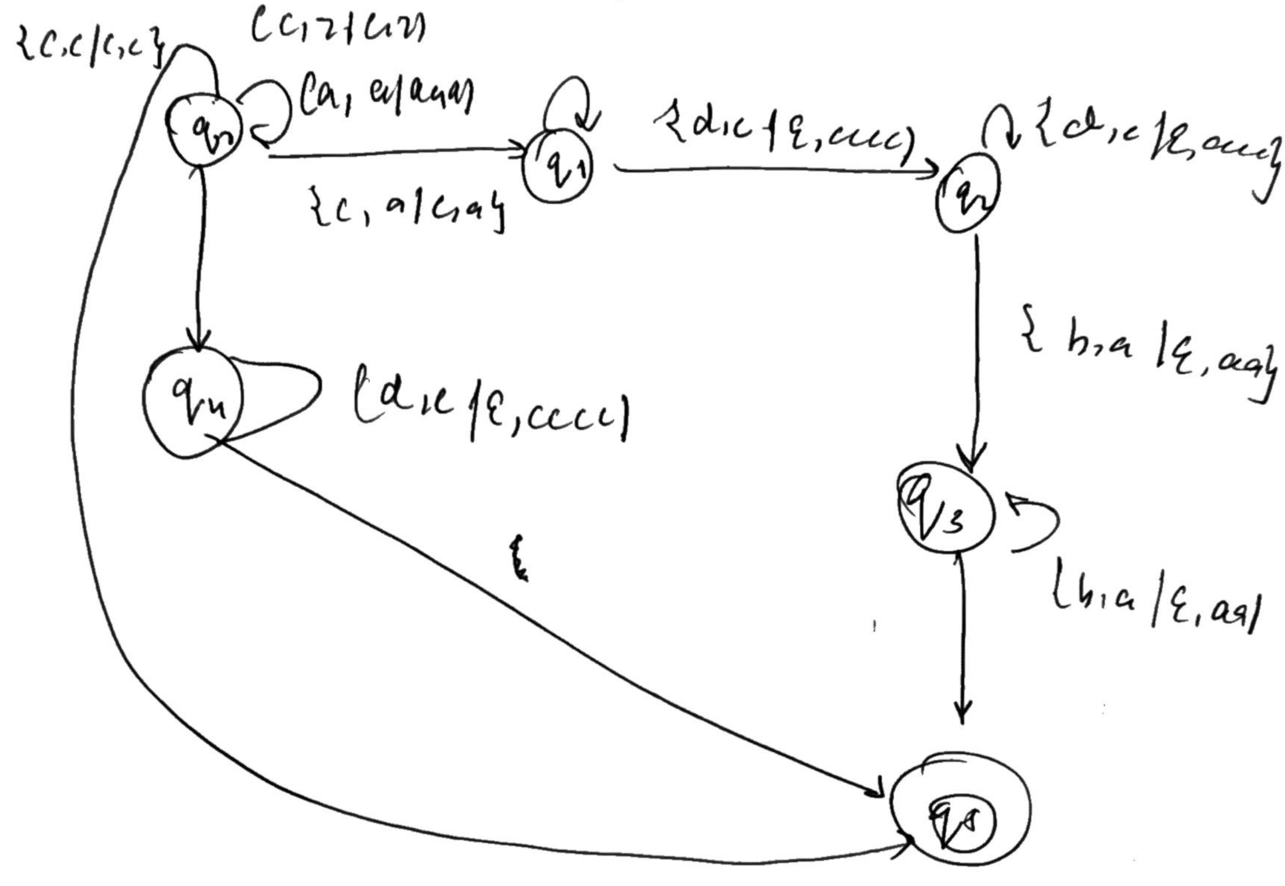
In this case the input string will be of the form $\{a^{(2*m)}b^m\}$. In this condition, keep on pushing a's in the stack until we encounter with b'. On recieving b check if top of stack is 'a', then pop 'aa' from the stack. Keep on popping aa's until all the b's of the string are processed. If we reach to the end of the input string stack becomes empty, then reached to the final state.

Case-3 m, n > 0 :-

In this cases the input string will be of the form $\{(a^{(2*m)}c^{(4*n)}d^n)^m\}$. Keep on pushing a's and c's on the stack until we encounted with d. Keep on popping cc's until all the d's of the string are processed. Then repeat the process in case 2 until reached to final or dead state.

Case - 4: min = 0 -

In this case the input string will be empty Therefore directly jump to final state.



(iv) (Case i)

Example:

rrgggg vy, rrrrgggggggg vvyy

(v) (Case ii)

Example:

rrgggg , rgg

(ii) Language for case 1

$L = \{ c^{(4+n)} d^{4} \}$

Language for case 2

$L = \{ a^{(2+n)} b^{n} \}$

(vi) Ambiguity in grammar

* A grammar is said to be ambiguous if there exists more than one leftmost derivation or more than one rightmost derivation or more than one parse tree for the given input string.

* If the grammar has ambiguity then it is not good for compiler construction. No method can automatically detect and remove the ambiguity.

RA1411030010044 (6)

As we can see in the above string there are more than 2 parse trees, so the grammar is ambiguous.

Q3

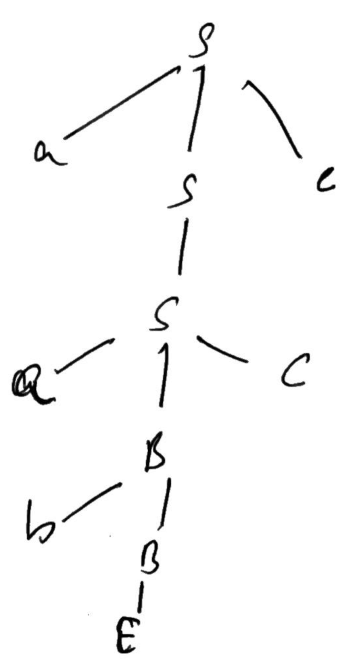
(a) $L = a^x b^y c^x$, let $x, y \geq 0$.

CFG:-

$S \rightarrow aSc \mid B \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

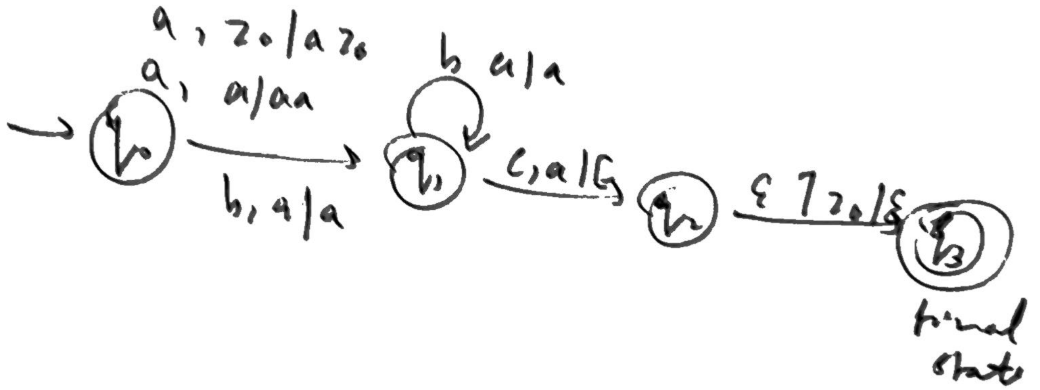
(d) Left most derivation for:- $a^2 b^1 c^2$



(1) $L = \{a^x b^y c^x\}$ RA1911003010094 (2)

Let $x, y \geq 1$

Let the string be $a^2 b^3 c^2$ = aabbbcc.



(2) Instantaneous Description :-

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

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

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

$\delta(q_1, c, a) = (q_2, \varepsilon)$

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

$\delta(q_2, \varepsilon, Z) = (q_3, \varepsilon)$ ← final state.

Turnstile Notation :-

for the string,

aabcc

Push
$$= \delta(q_0, aabcc, z_1) + (q_0, aabcc, z_0) + (q_0, abcc, az_0) + (q_1, bcc, aaz_0)$$ } No operation

Pop
$$+ (q_1, c, aaz_0) + (q_2, c, az_0) + (q_2, \varepsilon, z_0) + (q_3, \varepsilon)$$

(b) Rules for making a regular grammar

- It should have a non terminal in the left side
- have a right hand side that is
  - → ε, or
  - → a single terminal
  - → a single terminal followed by a single non-terminal.

But in our productions,

$S \rightarrow aSc \mid b \mid \varepsilon$.

$S \rightarrow asc$ and $S \rightarrow sB$

is violating the rules of the regular grammar. Hence, the language cannot be generated by the regular grammar.