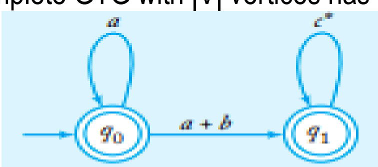
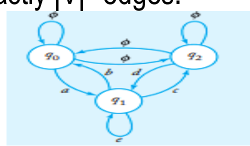
School of Computing
First CIA Exam –Aug 2024
Course Code: CSE301
Course Name: Theory of Computation
Duration: 90 minutes Max Marks: 50

Answer Key

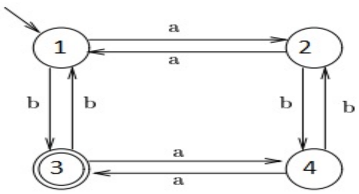
1. **Proof by induction**
   Basis: $l(0) = 1 = 2^0$. Root node, it has at most one leaf.
   Inductive: $l(i) \leq 2^i$ for $i=0,1, \ldots n$
   $l(n+1)= 2l(n)$
   $l(n+1) \leq 2 * 2^n = 2^{n+1}$. Thus, true for n, it must be true for n+1.

2. **Generalized transition graph** is a transition graph whose edges are labeled with regular expressions. The label of any walk from the initial state to a final state is the concatenation of several regular expressions, and hence itself a regular expression. A **complete GTG** is a graph in which all edges are present. A complete GTG with $|V|$ vertices has exactly $|V|^2$ edges.

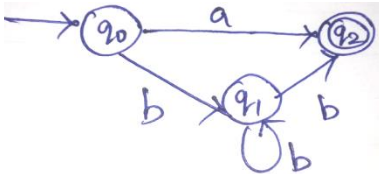




3. **dfa accepted even number of a's and odd number of b's**



4. **An nfa without λ-transitions and with a single final state for $\{a\} \cup \{b^n: n \geq 2\}$.**

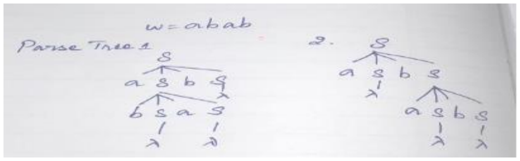


5. **Regular grammar $L=\{w : |n_a(w) - n_b(w)| \text{ is odd}\}$.**
   $S \rightarrow aA \mid bA$
   $A \rightarrow aS \mid bS \mid \lambda$

6. **Family of regular languages is closed under reversal.**
   - In a transition graph for a nfa, make the initial vertex a final vertex, and the final vertex the initial vertex, and reverse the direction on all the edges.
   - The modified nfa accepts w^R if and only if the original nfa accepts w. Therefore, the modified nfa accepts L^R, proving closure under reversal.

7. **Show that $L=\{a^n b^k c^{n+k} : n \geq 0, k \geq 0\}$ is not regular.**
   Use closure under homomorphism is easier to prove it. Take h(a)=a, h(b)=a, h(c)=c, then $h(L) = \{ a^{n+k} c^{n+k} : n+k \geq 0\} = \{ a^i c^i : i \geq 0\}$. Known proof is $a^i b^i$ is not regular; therefore, L cannot be regular either.

8. **Given grammar is ambiguous: $S \rightarrow aSbS \mid bSaS \mid \lambda$**



9. **Construct CFG for a C programming language "literal" and "for statement".**
   Literal → int | float | char | string

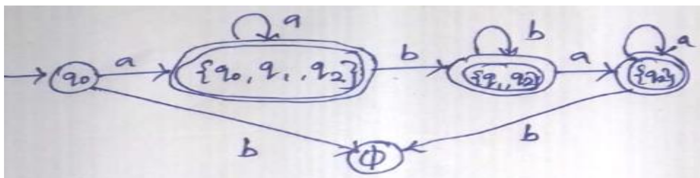
For → For <initialization>;<expression>;<increment / decrement><statement>
Initialization → declaration | assignment | λ

10. CFL for S → aaSbb | SS and S → λ

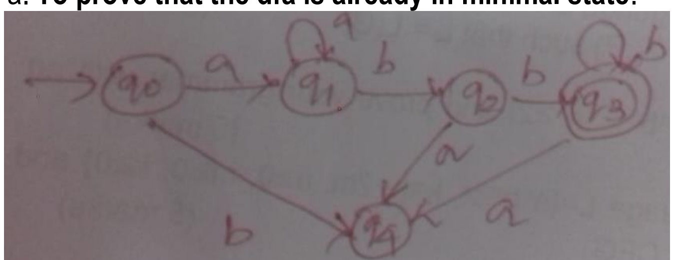
L={ w ∈ {aa, bb}* | where $|w_a| = |w_b|$ and for every prefix v of w, $|v|_a >= |v|_b$}.
Eg. $|v|_a = |v|_b$ => aabb and $|v|_a > |v|_b$ => aabbaabb (when compare to last bb, number of aa's are greater)

PART – B

11.nfa to dfa conversion



12. a. **To prove that the dfa is already in minimal state.**



(q_0, bb) ∈ q_4 – non final state
(q_1, bb) ∈ q_3 – final state
Therefore, q_0 and q_1 states are distinguishable
(q_0, b) ∈ q_4 – non final state
(q_2, b) q_3 – final state
Therefore, q_0 and q_2 states are distinguishable
(q_1, b) ∈ q_2 – non final state
(q_2, b) ∈ q_3 – final state
Therefore, q_1 and q_2 states are distinguishable

Hence, the given dfa is in minimal states.

b. **Construct right-linear grammar** with V = {q0, q1,… qn} and S= q0.

For each transition δ(qi, aj)= qk of M, the productions are
qi → aj qk.
If qk is in F, then add to P the production qk →λ.

Consider w ∈ L of the form w=ai aj … akal.
For M to accept this string it must make move on transition

$$\delta(q_0, a_i) = q_p,$$
$$\delta(q_p, a_j) = q_r,$$
$$\vdots$$
$$\delta(q_s, a_k) = q_t, \qquad q_0 \Rightarrow a_i q_p \Rightarrow a_i a_j q_r \stackrel{*}{\Rightarrow} a_i a_j \cdots a_k q_t$$
$$\delta(q_t, a_l) = q_f \in F. \qquad \Rightarrow a_i a_j \cdots a_k a_l q_f \Rightarrow a_i a_j \cdots a_k a_l,$$

With the grammar G, and w ∈ L(G).
This implies that δ*(q0, aiaj…..akal) = qf. Completing the proof.

13. a. **Find an s-grammar for L={$a^n b^{n+1}$: n ≥2}**

S → aA_1
A_1 → aA_2
A_2 → aA_2B_1 | bB_2
B1 → b
B2 → bB_3 and B_3→ b

i. Grammar (3 marks)
ii. Example (2 marks)

b. **Construct CFG for the language L={$a^n b^m c^k$, k=n+2m, n≥0, m≥0, k≥0}**

S → aSc | B | λ
B → bBcc | λ

i. Grammar (3 marks)

Example (2 marks)