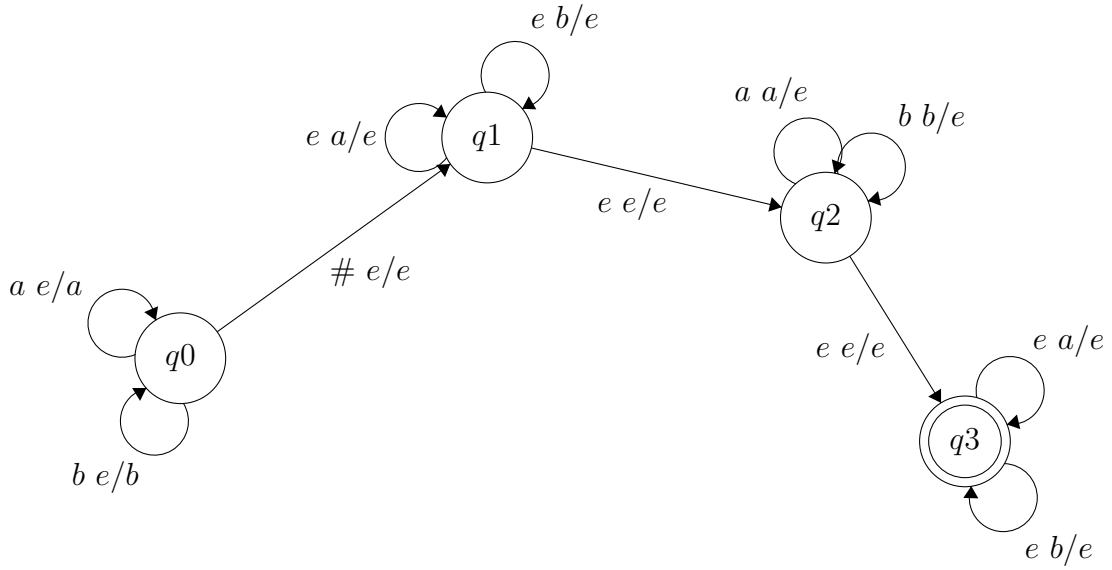
Student Information

Full Name : Furkan Karaca
Id Number : 2521698

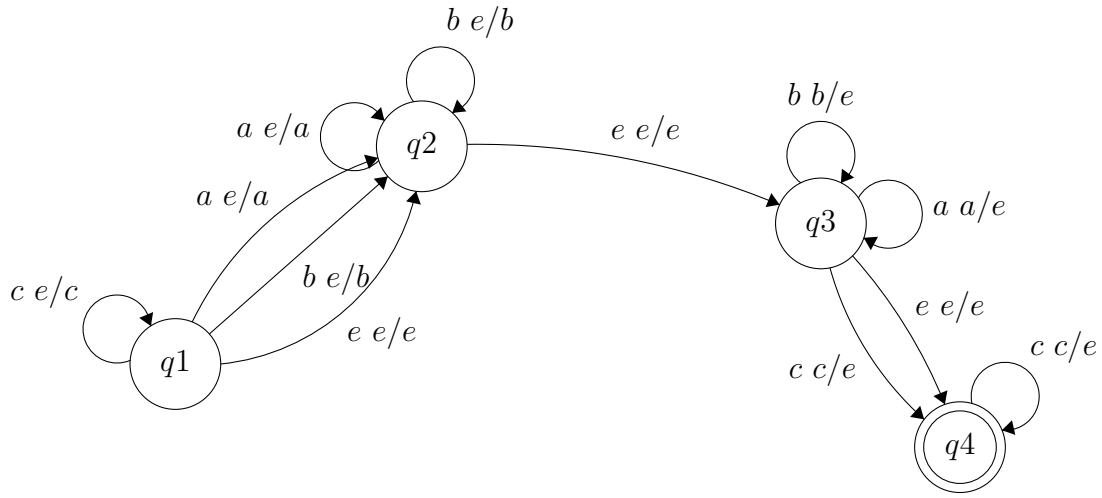
Answer 1

a)
$M = (K, \Sigma, \Gamma, \Delta, s, F)$
K=$\{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b, \#\}$
$\Gamma = \{a, b\}$
$\Delta = \{$
$\{(q_0, a, e), (q_0, a)\}$
$\{(q_0, b, e), (q_0, b)\}$
$\{(q_0, \#, e), (q_1, e)\}$
$\{(q_1, e, a), (q_1, e)\}$
$\{(q_1, e, b), (q_1, e)\}$
$\{(q_1, e, e), (q_2, e)\}$
$\{(q_2, a, a), (q_2, e)\}$
$\{(q_2, b, b), (q_2, e)\}$
$\{(q_2, e, e), (q_3, e)\}$
$\{(q_3, e, a), (q_3, e)\}$
$\{(q_3, e, b), (q_3, e)\}\}$
$F = \{q_3\}$
$s = \{q_0\}$



b)

K=$\{q_1, q_2, q_3, q_4\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{a, b, c\}$
$\Delta = \{\{(q_1, e, e), (q_2, e)\}$
$\{(q_1, c, e), (q_1, c)\}$
$\{(q_1, a, e), (q_2, a)\}$
$\{(q_1, b, e), (q_2, b)\}$
$\{(q_1, e, e), (q_2, e)\}$
$\{(q_2, a, e), (q_2, a)\}$
$\{(q_2, b, e), (q_2, b)\}$
$\{(q_2, e, e), (q_3, e)\}$
$\{(q_3, a, a), (q_3, e)\}$
$\{(q_3, b, b), (q_3, e)\}$
$\{(q_3, c, c), (q_4, e)\}$
$\{(q_3, e, e), (q_4, e)\}$
$\{(q_4, c, c), (q_4, e)\}\}$
$s = \{q_1\}$
$F = \{q_4\}$



Answer 2

Assume that our language is $\{0^n 1^n \wedge n \geq 1\}$. If we create our language as:
$S \longrightarrow 0F1$
$F \longrightarrow 0F1 \mid \epsilon$
Notice that our language cannot have an empty string. If we add two of these string combinations, it is still impossible to get an empty string that is in the $\{L\}^n$ but not in SS.

Answer 3

1.
L_1 can be recognized by our new automata. If we push a symbol when we read 'a' and stay in the same state, and when we read the first 'b' we move the final state and pop the symbol whenever we receive 'b'. We can make a trap state if we receive an 'a' in the second state. When the stack is empty and we are in the second state, we accept the string.
L_2 can be recognized by our automata. It is doable by using two different states and a bottom marker. One state is for the number of a is more than b and the second one is for b is more than a.
L_3 can be recognized by our automata. We should use only one symbol, and if we try to push symbols when we see 'a', pop when reading 'b' until the stack is empty, then continue pushing symbols until seeing a 'c', and pop one symbol for each 'c'. Since we use the bottom marker and know when to start pushing in the b state, it is doable.
2.
If we construct our language as $\{a^{2n}b^n c^n\}$ when we see 'a' we push and when we see 'b' we change state and pop from the stack, for c, we change state and pop from the stack again. This is our final state.
K=$\{q_1, q_2, q_3\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{t\}$

$\Delta = \{\{(q_1, a, e), (q_1, t)\}$
$\{(q_1, b, t), (q_2, e)\}$
$\{(q_2, b, t), (q_2, e)\}$
$\{(q_2, c, t), (q_3, e)\}$
$\{(q_3, c, t), (q_3, e)\}$
$\}$
$s = \{q_1\}$
$F = \{q_3\}$
{w , $a^k b^l c^l$, k=l+l}
S-¿aaSbc, e 3. It works like a counter.
4. When we push it increases its value and when we pop it decreases the value. But it cannot go to minus values and gives an error when exceeds zero. Also, we are not able to see its value whenever we want. We only know whether the counter is zero or not when we reach the end of a string in order to check its stack behavior. We can use it as a stack and extend our machine in order to get the desired S-CFL. Its operational semantics: Push = add 1, pop = subtract 1.
5. It is not closed under complementation since even a basic CFL is not closed under complementation, and our machine is weaker than a CFL, which means we cannot transform our machine into a DFA-recognizable machine. L_1 is an example of it. The complement of L_1 is not recognizable in our machine.