CS 310, Assignment 3

Answers

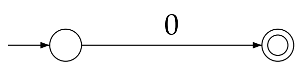
1. Using the method described in class and in the textbook (Section 9.1) convert the following regular expression into a state transition diagram:

$$(10^*1 + 01^*0)^*$$

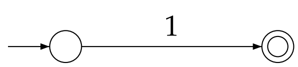
Indicate in your answer how did you arrive at the result as follows: Write down all the state transition diagrams that you constructed for all the subexpressions and clearly indicate which diagram corresponds to which expression. Do *not* simplify any state transition diagram.

ANSWER:

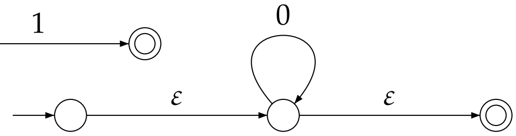
(a) Automaton for 0:



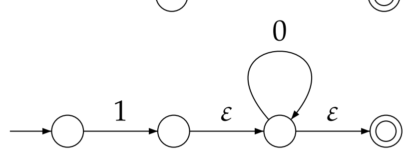
(b) Automaton for 1:



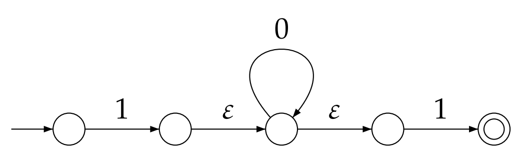
(c) Automaton for 0^* (from 1a):



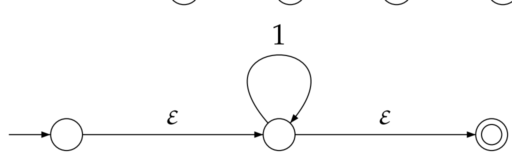
(d) Automaton for 10^* (from 1b and 1c):



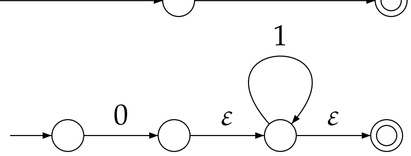
(e) Automaton for 10^*1 (from 1b and 1d)



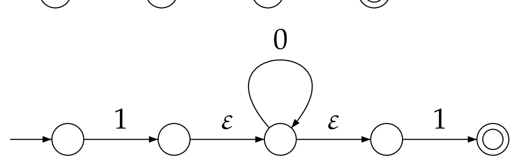
(f) Automaton for 1^* (from 1b):



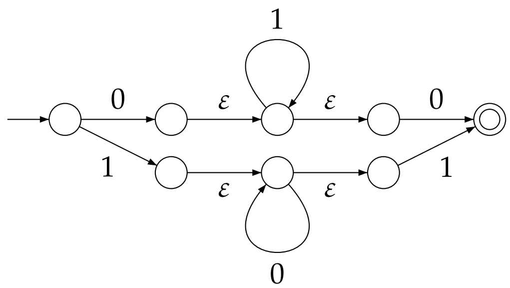
(g) Automaton for 01^* (from 1a and 1f):



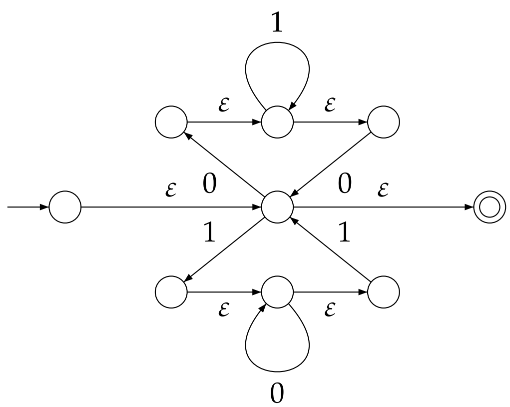
(h) Automaton for 01^*0 (from 1a and 1g)



(i) Automaton for $10^*1 + 01^*0$ (from 1e and 1h):



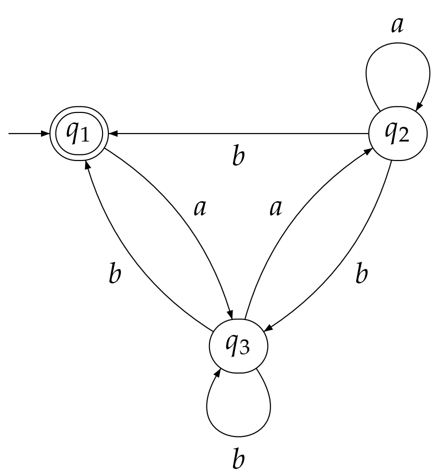
(j) Automaton for $(10^*1 + 01^*0)^*$ (from 1i, the answer to the question):



Note that I used the technique that merges states. Using ε-transitions instead would have been equally fine, but the resulting automaton would have been considerably larger.

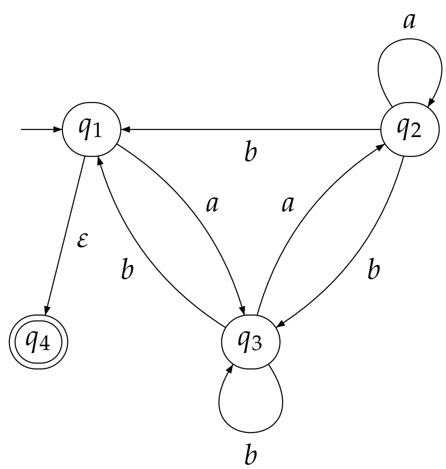
---

2. Consider the following state transition diagram over $\Sigma = \{a, b\}$:

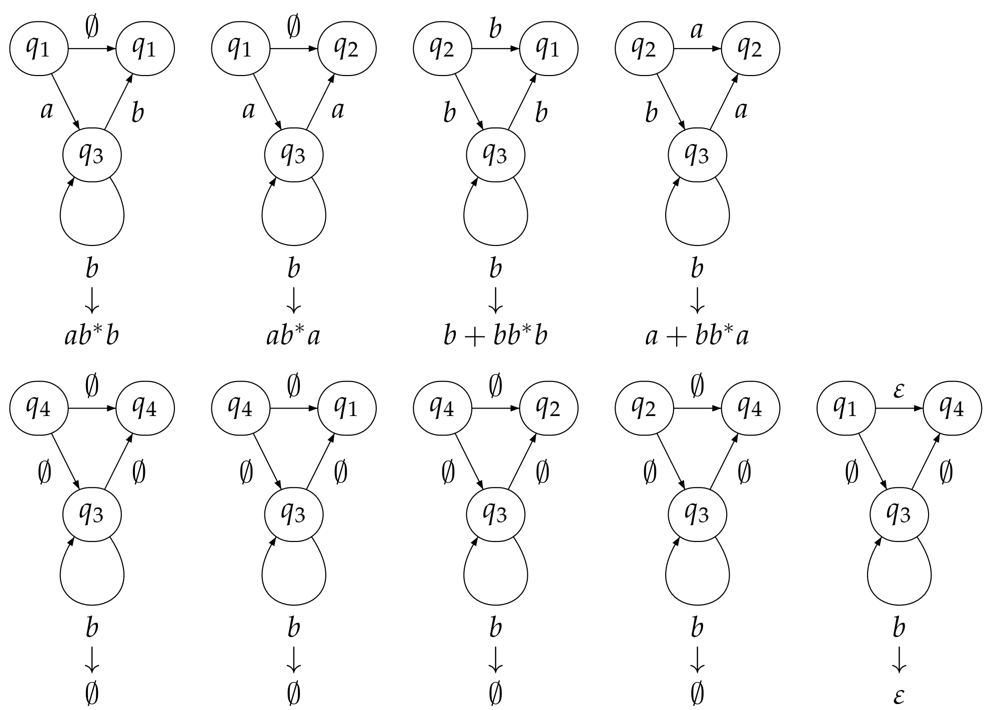


Using the method described in class and in the textbook (Section 9.2) convert the diagram into an equivalent regular expression. Include all the intermediate steps in your answer.

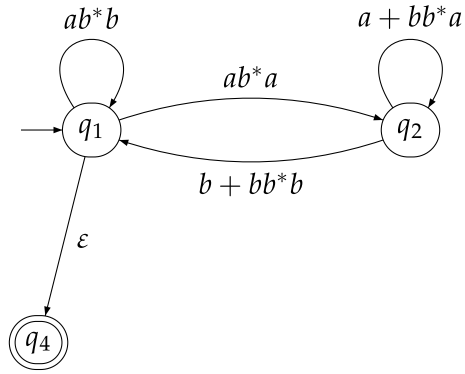
ANSWER: We have a single accepting state which happens to be identical to the initial state. We can live with that and eliminate all the other stated, ending with a single state and a loop transition as shown in class. I am going to do it the hard way though (following the algorithms from the textbook) and separate the initial and accepting state by introducing a new accepting state:



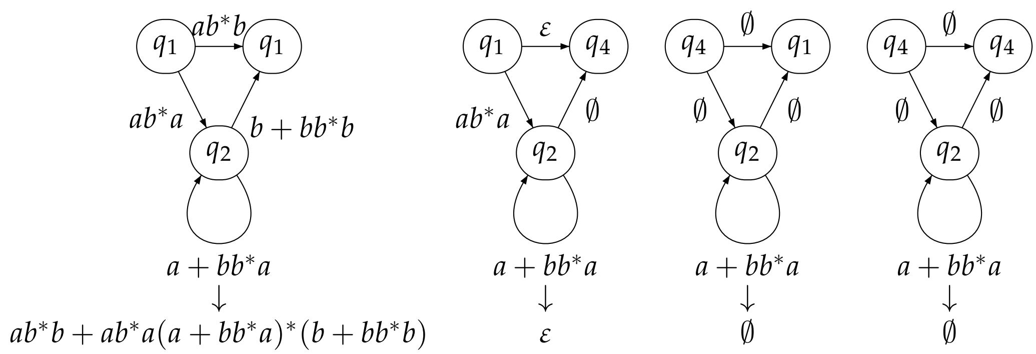
We now have two states that need to be eliminated. We start by eliminating q_3 (just a random choice, we could have eliminated q_2 instead). We have the following "triangles" with their respective regular expressions:



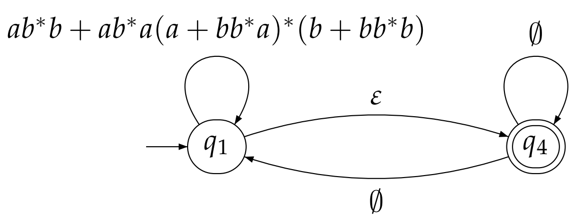
This in turn results in the following generalized state transition diagram:



We then eliminate the remaining state which is neither initial nor accepting namely, q_2:



We end up with the following generalized transition diagram:



The regular expression equivalent to the original transition diagram is therefore:

$$(ab^*b + ab^*a(a + bb^*a)^*(b + bb^*b))^*\varepsilon(\emptyset^* + \emptyset(ab^*b + ab^*a(a + bb^*a)^*(b + bb^*b))^*\varepsilon)^*$$

Given that $\emptyset^* = \varepsilon$, $\varepsilon^* = \varepsilon$, ε is an identity for concatenation, and $\emptyset$ is a zero for concatenation the expression can be easily simplified to the following:

$$(ab^*b + ab^*a(a + bb^*a)^*(b + bb^*b))^*$$

3. Are the languages L_1 and L_2 below over the alphabet $\Sigma = \{a, b, c\}$ regular or non-regular? Justify your answer carefully.

   (a) $L_1 = \{a^{2i+1}b^j : i, j \geq 0\} \cap \{a^k b^{2n+1} c^{3p} : k, n, p \geq 0\}$

   ANSWER: Arguably the simplest way of showing that L_1 is regular is to note that $L_1 = (L_{11}L_{12}) \cap (L_{13}L_{14}L_{15})$, where $L_{11} = \{a^{2i+1} : i \geq 0\} = (aa)^*a$, $L_{12} = \{b^j : j \geq 0\} = b^*$, $L_{13} = \{a^k : k \geq 0\} = a^*$, $L_{14} = \{b^{2n+1} : n \geq 0\} = (bb)^*b$, and $L_{15} = \{c^{3p} : p \geq 0\} = (ccc)^*$.

   All the languages L_{1j} are regular, $1 \leq j \leq 5$; indeed, I just gave above the respective regular expressions. Thus L_1 is a combination of concatenating and intersecting regular languages (see above), and regular languages are closed under both concatenation and intersection. It follows that L_2 is regular. □

   A more direct way to show the same thing is to note that L_1 cannot contain any c (since c's do not appear in the first term of the intersection) and so a string in L_1 has the form $a^x b^y$ for some $x, y \geq 0$. Furthermore x must be odd (because of the first term in the intersection) and y must also be odd (this time because of the second term). In other words, $L_1 = \{a^{2i+1}b^{2n+1} : i, n \geq 0\} = (aa)^*a(bb)^*b$. Thus L_1 is regular. □

   (b) $L_2 = \{a^i b^{j+2} c^{2i} : i, j \geq 0\}$

   ANSWER: L_2 is not regular and we will prove it so using the pumping lemma.

   Assume therefore that L_2 is regular. Note that both i and j are arbitrarily large, so there exists a string $w = a^i b^{j+3} c^{2i} \in L_1$ that is longer than the threshold n and so the pumping lemma applies to it. In fact we will take $i = n$ (i.e., w is *much* longer than n).

   From the pumping lemma we have that $w = xyz$ such that $xy^2z \in L_2$. We furthermore have $|xy| < n = i$. It follows that y only contains a symbols that is, $y = a^m$ for some $m > 0$ (indeed, $|xy| < i$ so xy must come from the first i symbols of w which are all a's).

   If this is the case, then xy^2z has $i + m$ a's (we have increased their number since we pumped y) and $2i$ c's (we have not touched those). Since $xy^2z \in L_2$ it follows that the numbers c's is twice as much as the number of a's that is, $2i = 2(i + m)$ which is equivalent to $i = i + m$ and so $m = 0$. This contradicts the fact that $m > 0$ and so our initial assumption (that L_2 is regular) must be false. □

   What happens with the b's you ask? We have not touched those simply because we were able to come up with a string w long enough so that the b's do not enter the picture. This is fortunate, since the b's could have been pumped liberally and so would have spoiled our day.
