Homework #1

Artemis Kelly
CS424 - Theory of Computation

January 12, 2022

Question 1.6

Give state diagrams of DFAs recognizing the following languages. In all parts, the alphabet is {0,1}

Question 1: 1.6b

Question: {w ‖ w contains at least three 1s}

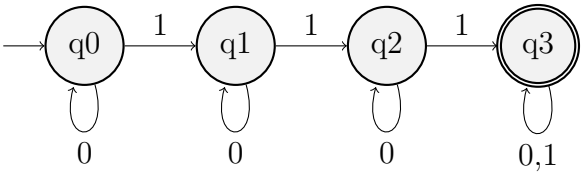

Figure 1: DFA

Question 2: 1.6c

Question: {w ‖ w contains the substring 0101 (i.e., w = x0101y for some x and y)}

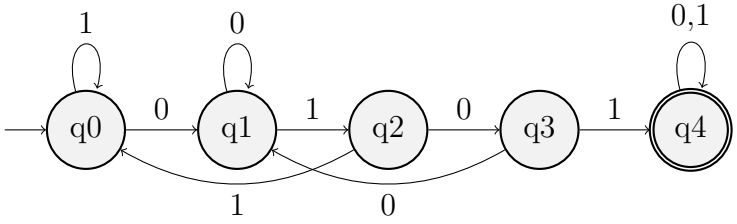

Figure 2: DFA

Question 3: 1.6f

Question: {w ‖ w doesn't contain the substring 110}

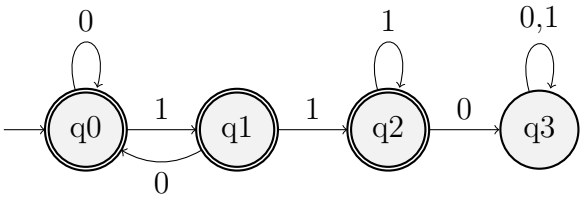

Figure 3: DFA

Question 1.8

Question Use the construction in the proof of Theorem 1.45 to give the state diagrams of NFAs recognizing the union of the languages described in.

Question 4: 1.8b

Exercises 1.6c and 1.6f.

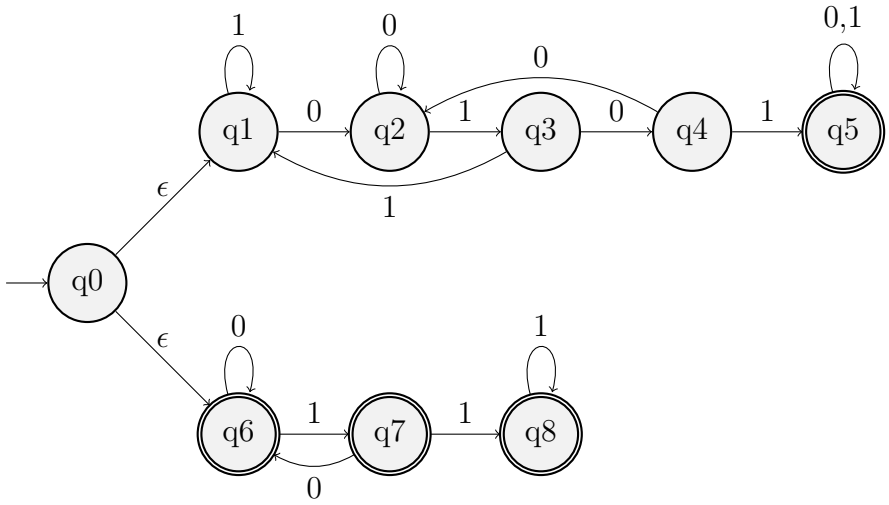

Figure 4: NFA Combiation

Question 5: 1.10

Question: Use the construction in the proof of Theorem 1.49 to give the state diagrams of NFAs recognizing the star of the languages described in

Question 1.10a

Exercise 1.6b

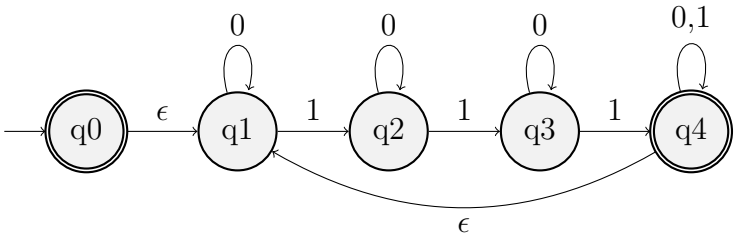

Figure 5: NFA Recognizing language of 1.6b

Question 6: 1.16

Use the construction given in Theorem 1.39 to convert the following two non deterministic finite automata to equivalent deterministic finite automata.

Question 1.16b

Question: The NFA shown in the book has 3 states. Therefore, the DFA we can construct, which will accept the same language, must have 2^3 states.

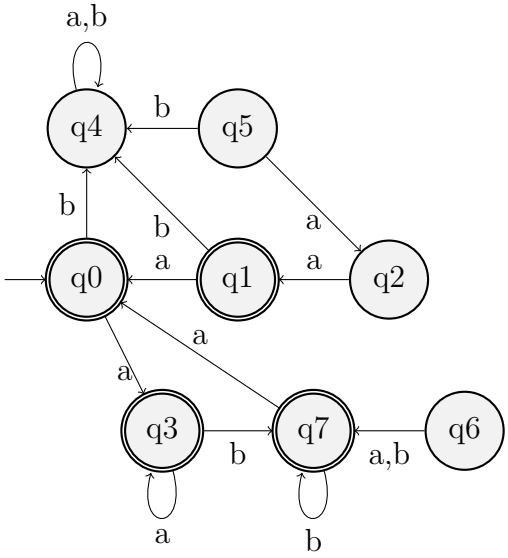

Figure 6: NFA Recognizing language of 1.6b

Question 7: 1.21

Question: Use the procedure described in Lemma 1.60 to convert the following finite automata to regular expressions.

Question 1.21b

Answer: The regular expression for the finite automata is ϵ|(a|b)(a|bb|ba(a|b))*(b|ba). This is found by adding an additional start state with an ϵ transition to the beginning of the finite automata and then going through to get rid of states one at a time until you are left with a single transition, a start state and an accepting state.

Question 8: 1.30

Question: Describe the error in the following "proof" that 0*1*is not a regular language. (An error must exist because 0*1*is regular.) The proof is by contradiction. Assume that 0^*1^* is regular. Let p be the pumping length for 0^*1^* given by the pumping lemma. Choose s to be the string 0^p1^p. You know that s is a member of 0^*1^*, but Example 1.73 shows that s cannot be pumped. Thus you have a contradiction. So 0^*1^* is not regular.

Answer: Lets call the language described above L. Looking at example 1.73, 1 and 2 are not relevant. The error in this proof is that using a single string as a counter example to show one language is non regular does not mean that every language which accepts that string is non regular. While the string xz would contain fewer zeros than ones, that is still in the language L because we do not need the same amount of zeros or ones.

Question 9: 1.36

Question: Let Bn = $\{a^k \parallel$ k is a multiple of n$\}$. Show that for each n$\geq$ 1, the language Bn is regular.

Answer: We build a DFA with n states, q_0, q_1 ,... q_{n-1}. These will serve to count the number of consecutive a's modulo n (because for each a that is input, the "counter" increases by 1 as it moves to the next state). q_0 is the only accepting state, at which point there must be n a's, in which case k must be a multiple of n. If it is not, then the DFA will end in a non accepting state. Because there is a DFA which recognizes this language, it must be a regular language.

Question 10: 1.42

Question: For languages A and B, let the shuffle of A and B be the language. $\{$w $\parallel$ w = a_1b_1 ... a_kb_k, where a_1 ... $a_k \in$ A and $b_1...b_k \in$ B, each a_i,$b_i \in \Sigma^*$ $\}$. Show that the class of regular languages is closed under shuffle.

Answer: To show using languages A and B that the class of regular languages is closed under shuffle, A and B must be regular languages. Therefore, there is a DFA that can describe each of them. Lets take DFA_A to be (Q_A, Σ, δ_A, q_A, F_A) and DFA_B to be (Q_B, Σ, δ_B, q_B, F_B). We will use DFA S to be (Q, Σ, δ, q, F), to describe the shuffle of A and B.

1) Q = Q_A x Q_B U q_0, Q_A x Q_B gives us all of the possible states of DFA_A and DFA_B and q_0 is the start state

2) q = q_0

3)δ must describe that if the current state is in DFA_A, then the next input symbol must be in DFA_B and vice versa. We only need to change the current state of the DFA which is reading an input symbol. So δ((x,y,A)a) = (δ_A(x,a),y,B) and vice versa δ((x,y,B)b) = (x, δ_B(y,a)A) δ must work with transitions for 3 different scenarios. First, $\delta(q_0, \epsilon) = (q_A,q_B)$. This means that at the start state, both DFA_A and DFA_B can go to their start states without reading anything in. Second, (δ_A(x,a),y) $\in\delta$((x,y),a). This means that if we read in a character that should move DFA_A to the next state, we can do so accordingly without changing the state of DFA_B. Lastly, we have the opposite in which case the next input character is used to change the state of DFA_B and not DFA_A. (x,δ_B(y,a)) $\in\delta$((x,y),a).

4) S will accept the input string if both DFA_B and DFA_A are in the accept states. It will also accept the empty string, at which case the input will have ended with S in its start state F = F_A x F_B U q_0

The DFA which recognizes the shuffle can alternate between an input character recognized by DFA_A to an input character recognized by DFA_B as each character is read in, but does not have to. Therefore, S must keep track of which state DFA_A is in and which state DFA_B is in so that when it takes it's next input character, it can see which DFA to move accordingly. When it recognizes which DFA's state to switch, it must switch appropriately. When the entirety of the input is processed, if both DFA_A and DFA_B are in accept states, then the input string is accepted and S can accept. Because there is a DFA that describes the regular language of the shuffle of regular languages A and B, the shuffle itself is also a regular language.