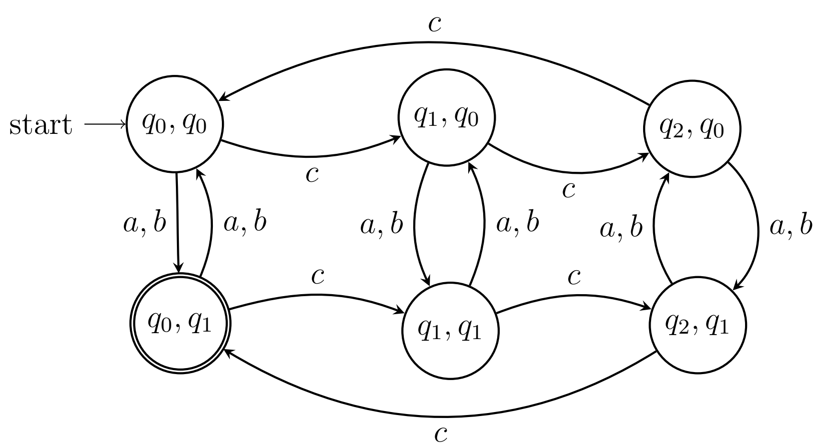
2AC3 Assignment 1

Maxwell Huang

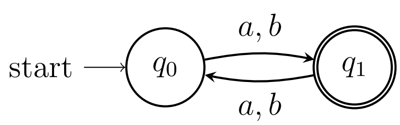
February 8th, 2025

1 Introduction

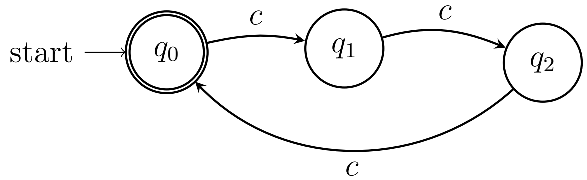
1.1 Question 1



We can break down Question 1's requirements into 2 simple DFAs. First, the number of a's + b's must be odd. Drawing this DFA is quite simple, we can simply have 2 states, where transitions a or b will move from one state to the other like so:



The second DFA involves the number of c's to be divisible by 3. We can simple create a dfa with 3 states (q0, q1, q2) and c moves you to the next state:



We simply have to cross-product the 2 DFAs above, which includes multiplying their sets of states Q1 and Q2, their final states F1 and F2, and combining their traversal functions. This results in the function at the top with the only final state being q0 from our second DFA meaning c is divisible by 3 and q1 from our first DFA, meaning the number of a's and b's is odd

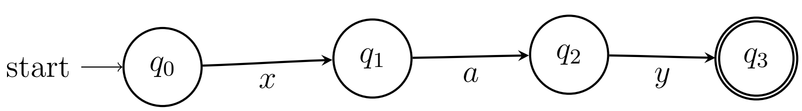
1.2 Question 2

Again, let's break down our NFA. The requirements state that the strings accepted by the NFA N2 are xay, where x and y are both strings accepted by NFA N1. The formal definition of our NFA is as follows (Where Si = sigma, D1 = delta1, S1 = Set of start States): N1 = (Q1, Si, D1, S1, F1)

N2 = (Q1 + qn + Q1, Si, D2, S1, F2)

Let's work through the NFA N2. Our Set of states is going to be Q1's states, then an intermediate state, followed by another set of Q1's states. This is simply because we accept all xay, where x and y are both part of N1. Our language of acceptable transitions Si is the exact same thing as N1. Next, our Transition function is simply an updated version of N1's, which considers the additional states from our second set of states Q1's, and the qn in the middle. Our starting set should be the exact same as N1 since the first section of an acceptable string for N2 must be accepted by N1. Finally F2 is simply the accepted states in our section set of states Q1.

Here's a rough NFA for N2, where q1 and q3 are both NFAs of N1:



1.3 Question 3

The NFA we would need here is extremely simple. L(N) is a NFA or language that accepts all x's where x contains a substring of EITHER DFA M1 or M2.

This means our NFA for L(N) can simply have a single start node with an epsilon transition to 2 different sections. The first section will be identical to DFA M1 and the second section will be identical to DFA M2 with one small change. All final states will also all transition to themselves since x only must contain a string accepted by M1 or M2.

This works since this will accept ALL strings that have some substring in M1 or M2. Since any substring of M1 or M2 will be accepted, this NFA doesn't accept any strings which L(N) normally wouldn't.

Here's our formal definition: M1 = (Q1, Si, d1, s1, F1) M2 = (Q2, Si, d2, s2, F2) N = (Q1 U Q2 U s3, Si + Eps, D1, s3, F1 U F2)
Our states for N are all the states of Q1, and Q2 as well as our new starting state.

Our alphabet is the exact same as the DFAs M1 and M2, except for the addition of the epsilon transition

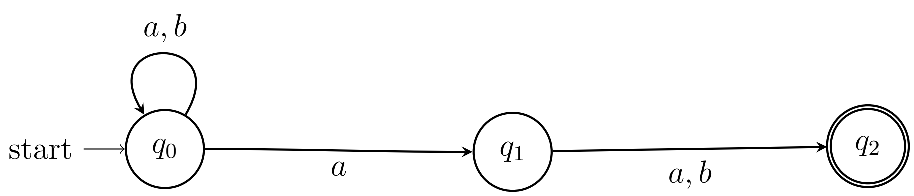
Our transition function is a combination of d1 and d2, as all the states from both DFAs M1 and M2 are here. There's also the added transitions for the final states and the epsilon transition from s3 to s1 and s2.

Our starting state is s3 which is a state that epsilon transitions to both s1 and s2.

Finally, our final states are the union of F1 and F2.

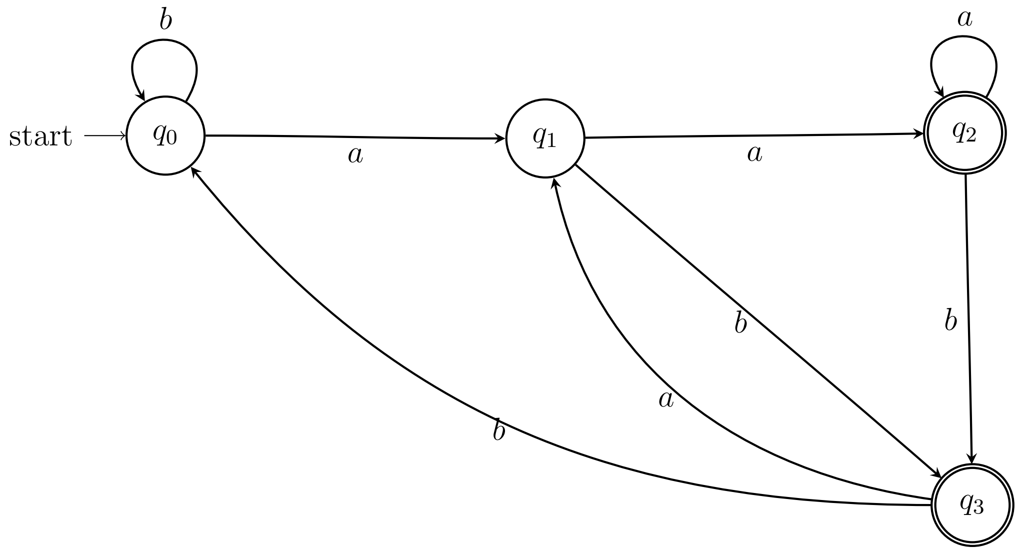
1.4 Question 4

Part A)



Part B) The process for turning this into a DFA isn't too complicated. In this example, if we have an a in the middle of the string we don't need to worry about keeping track if the next character is an a or b because the self loop at the start guarantees only the last 2 characters of any given string matter.

For the DFA once we reach q1 we need to determine if the next character is an a or a b. So we add another final state and put transition b there. Then we need to add transitions to q2 and our new q3. Here's the final result:


start
q_0
q_1
q_2
q_3
a
b