CSCI 510 Models of Computation

HW6: Still Alive

Solution Key

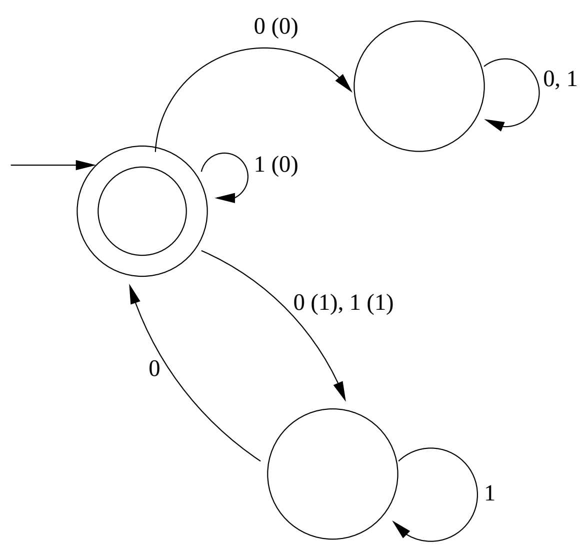
Include your full name*, CS login, and the problem number(s) on each piece of paper you hand in, and please staple your pages together before handing in.*

While collaboration is encouraged in this class, please remember not to take away notes from collaboration sessions other than your scheduled lab section.

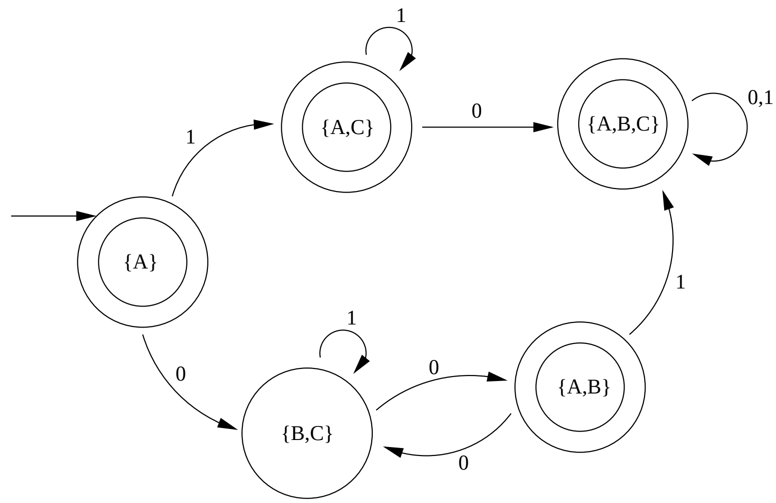
In general, if you submit a complicated, messy FSM and include no explanation of how it works, it will probably not be graded.

Problem 1

Convert the following NFSM to a DFSM:

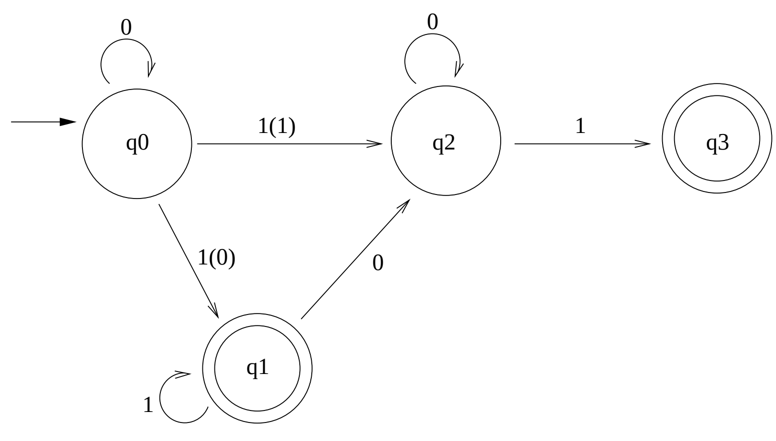


Label the states A, B, and C, with A being the start state, B being the top-left state, and C being the bottom-left state.



Problem 2

Write down the regular expression for the language that the following NFSM recognizes:



The following regular expression represents the NFSM:

$$0^*1\left[1^* + (\varepsilon + 1^*0)\,0^*1\right]$$

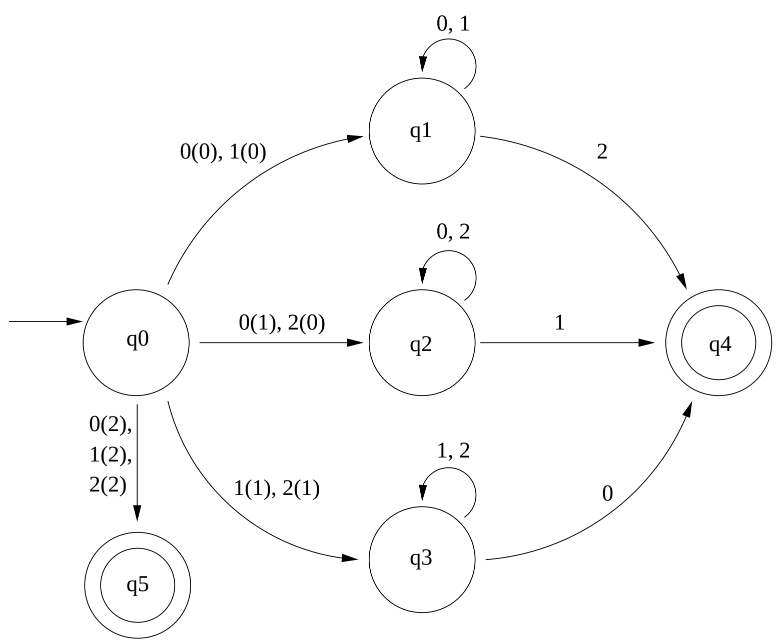
This can be simplified further to $0^*1^*0^*1$.

Problem 3

Let $\Sigma = \{0, 1, 2\}$, and let L be the language over Σ that contains each string w ending with some symbol that does not occur anywhere else in w. For example, 011012, 11120, 0002, 10, *and* 1 *are all strings in L.*

a. Construct a nondeterministic finite-state machine that accepts L.

b. Give a regular expression describing the language L.



a.

b. $[(0+1)^*2] + [(0+2)^*1] + [(1+2)^*0]$

Problem 4

Describe an algorithm that constructs an NFSM N with input alphabet Σ from a regular expression r over Σ such that N recognizes a string w if any substring of w is in the language defined by r. In other words, N recognizes the language

$$L = \{w = xyz \mid y \text{ is in the language defined by } r \text{ and } x \in \Sigma^* \text{ and } z \in \Sigma^*\}.$$

Let R be an NFSM that recognizes the language defined by r. We construct N as follows. On the start state q_0 of R, add one transition from q_0 to itself for each character in Σ in addition to any existing transitions. Then, for each q in the set of accept states F of R, add a transition from q to itself for each character in Σ, in addition to any existing transitions.

This works because if $w = xyz \in L$, then N will loop on itself at q_0 for each character in x, then compute normally for y, getting to an accept state, and then loop on that accept state for z. This means it will end on an accept state, so N will accept w.

If N accepts w, then we can split w into xyz where x is the largest prefix of w such that N ends at q_0 on input x. So, N can simply loop on q_0 for all of x and get the same result. Then y is the shortest string such that xy is a prefix of w and N accepts y. We know such a string exists because N accepts w and it is in its starting state after computing x, so it must get to an accepting state after that, so y must exist (even if it is empty). Then z is the rest of the w. Since N accepts y, we know that y is in the language defined by r, so $xyz = w \in L$.

Hence we see that N recognizes L.