

## Solutions to Homework Assignment 4

### 1. Part (a):

**Alphabet:**  $\{a, b\}$

**Set of states:**  $\{q_0, q_1, q_2, q_3\}$

**Initial state:**  $q_0$

**Set of final states:**  $\{q_1, q_3\}$

**Transition relation:** The following transitions are allowed:  $\langle q_0, a, q_1 \rangle$ ,  $\langle q_0, b, q_2 \rangle$ ,  $\langle q_1, b, q_2 \rangle$ ,  $\langle q_1, a, q_3 \rangle$ ,  $\langle q_2, a, q_2 \rangle$ , and  $\langle q_2, b, q_3 \rangle$ .

### Part (b):

**A  $FS_1$  computation of  $baaa$ :**

Step	State	Scanning Position
0	$q_0$	$\langle \epsilon, baaa \rangle$
1	$q_1$	$\langle b, aaa \rangle$
2	$q_2$	$\langle ba, aa \rangle$
3	$q_2$	$\langle baa, a \rangle$
4	$q_2$	$\langle baaa, \epsilon \rangle$

**A  $FS_1$  computation of  $abb$ :**

Step	State	Scanning Position
0	$q_0$	$\langle \epsilon, abb \rangle$
1	$q_1$	$\langle a, bb \rangle$
2	$q_2$	$\langle ab, b \rangle$
3	$q_2$	$\langle abb, \epsilon \rangle$

**Part (c):** For each state of  $FS_1$  and symbol of the alphabet, there is at most one arc labeled with the symbol leading from the state. This means that there can be at most one  $FS_1$  computation for each string of the alphabet. The computation for  $aaa$  in  $FS_1$  is this:

Step	State	Scanning Position
0	$q_0$	$\langle \epsilon, aaa \rangle$
1	$q_1$	$\langle a, aa \rangle$
2	$q_2$	$\langle aa, a \rangle$

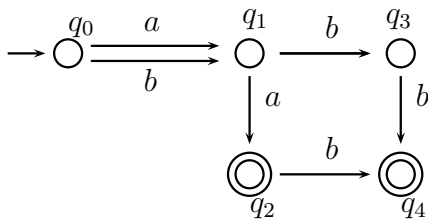
The computation can't go beyond Step 2 because there is no transition for  $b$  from  $q_3$ . This computation is not successful because it doesn't reach the end of the string. Since there are no other computations of  $bbb$ , this string is not accepted by  $FS_1$ .

**Part (d):**  $FS_1$  accepts strings of this type:  $a$ ,  $aa$ ,  $ab$ [any number of  $a$ 's] $b$ , and  $b$ [any number of  $a$ 's] $b$ .

Here is how we can write the set of these strings as a regular language:

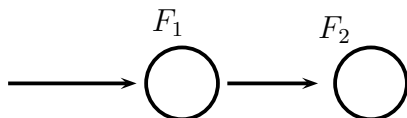
$$(\{a\} \circ \{b\} \circ \{a\}^* \circ \{b\}) \cup (\{b\} \circ \{a\}^* \circ \{b\}) \cup (\{a\} \circ \{a\}) \cup \{a\}$$

2. Here is a machine that accepts just  $aa$ ,  $ba$ ,  $aab$ ,  $bbb$ ,  $abb$  and  $bab$ :



3. Say that  $X$  and  $Y$  are sets of strings accepted by FSA's  $FS_X$  and  $FS_Y$  respectively. Explain how you would design an FSA that will accept exactly the set  $X \circ Y$ .

This problem can be done by treating  $F_X$  and  $F_Y$  as separate modules we glue together, though a bit of care has to be taken with the transition from states of  $F_X$  to states of  $F_Y$ . The flowchart we want is simple enough:



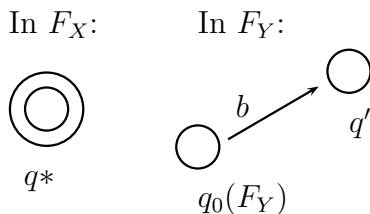
We want those computations that result from running  $F_X$  until a final state, and then moving to  $F_Y$  to run the rest of the tape.

The one wrinkle is that when you enter a final state - call one  $q^*$  - of  $F_X$ , you are in a state that would be the end of a computation accepting a string consisting of the letters you have read up to entering  $q^*$ . Then if there are further letters, you want to read them with  $F_Y$ , to see if the rest of the letters are a string  $F_Y$  will accept. But you can't just read the next letter and go into the initial state of  $F_Y$ , since a normal  $F_Y$  computation is in the initial state after reading *no* letters, not after reading one letter. So we can't just go from the final states of  $F_X$  to the initial state of  $F_Y$ ; we need a little more finesse.

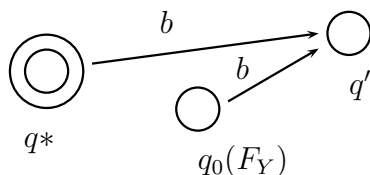
How do we get this to work? One way is to use the "jumps" discussed in Problem 1 on page 134. Define a machine that - when in a final state of  $F_X$ , jumps to the initial state of  $F_Y$  without reading anything. I'll take a different path, avoiding jumps.

Here is how to splice the machines together. Call the new machine  $F_{splice}$ . The states of  $F_{splice}$  are the states of  $F_X$  and  $F_Y$ . The transitions of  $F_{splice}$  are the transitions of  $F_X$  and  $F_Y$  with these extras: Take any final state  $q^*$  of  $F_X$ . For every letter  $u$ , look to see if  $F_Y$  does anything if it reads  $u$  in the initial state. If  $F_Y$  does nothing, don't add any transition from  $q^*$  on reading  $u$ . If  $F_Y$  moves into state  $q'$ , then add the transition  $\langle q^*, u, q' \rangle$  to the transitions of  $F_{splice}$ . **NOTE: This is why I had to allow the machine to be nondeterministic. This step won't produce a deterministic machine if there is already a transition  $\langle q^*, u, q^{**} \rangle$  in  $F_X$  for some state  $q^{**}$  of  $F_X$ .**

In pictures, say we have this situation (where  $q_0(F_Y)$  is the initial state of  $F_Y$ ):



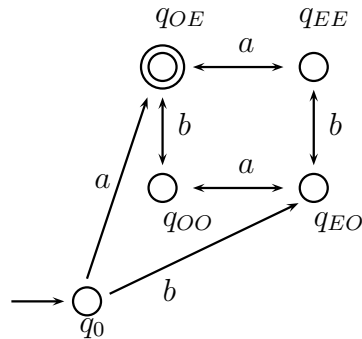
We need  $F_{splice}$  to look like this:



The final states of  $F_{splice}$  are the final states of  $F_Y$ . If the initial state of  $F_Y$  is a final state, then all the final states of  $F_X$  are final states  $F_{splice}$  as well. The initial state of  $F_{splice}$  is the initial state of  $F_X$ .

4. Devise a deterministic Finite State Automaton over the alphabet  $\{a, b\}$  that accepts all and only strings which have an odd number of  $a$ 's and an even number of  $b$ 's.

It will be easier to keep track of what is going on if we choose suggestive names for the states. We'll use  $q_{OO}$  to be the state registering an odd number of  $a$ 's and an odd number of  $b$ 's, with  $q_{OE}$  the state for an odd number of  $a$ 's and an even number of  $b$ 's,  $q_{EO}$  the state for an even number of  $a$ 's and an odd number of  $b$ 's, and  $q_{EE}$  the state for an even number of  $a$ 's and an even number of  $b$ 's. This machine will do what we need:



(To keep the diagram simpler I'm cheating a little bit by using two-headed arrows to indicate arrows in both directions.) This machine is a little inefficient, for the sake of clarity. We don't actually need a separate start state  $q_0$ . Instead we could combine  $q_0$  and  $q_{EE}$  into a single state.

**An aside about clear thinking.** This problem and solution illustrate an important informal observation about representing problems in a way that facilitates problem-solving. Sometimes you make an important step toward success just by intelligently choosing names for the things you are talking about. So in this case, it makes sense to write:  $q_{EE}, q_{EO}, q_{OE}, q_{OO}$ . Once you present the states you need this way, the way to write in transitions (draw arrows) and final states is pretty obvious.