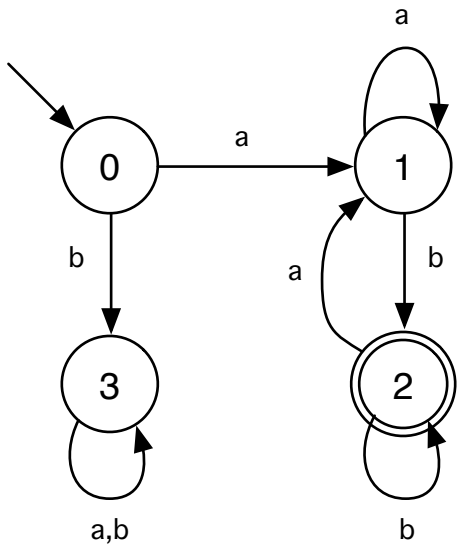
Regular Languages Homework

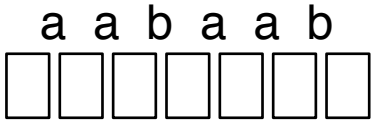
WU CS 465—Fritz Ruehr—Spring 2019

These questions on regular expressions (REs) and (deterministic) finite automata (DFAs) are intended to be straightforward, with a bit of effort—you shouldn't need any sophisticated techniques to answer them, although there is just a bit of creative/"aha!" work in designing machines.

1. Consider the following DFA, drawn in the usual style (i.e., initial-state arrow coming in from upper-left, doubled circles for final states, etc.):



(a) show that the string "`aabaab`" is in the language of this machine by marking the states that it transitions through below the letters in the box diagram below (start with the initial state 0 on the left; the last state on the right should be final):



(b) write out the transition table description for this machine (put the states down the right side and the letters (a and b) across the top; put the next state in the "cells" of the table):

(c) give a regular expression description of this machine, i.e., an RE which is equivalent in terms of accepting the same language:

2. Consider the usual alphabet of just the two letters {a,b}; give both a regular expression and a DFA (machine) to describe the language over this alphabet with only strings of an even length (i.e., length 0, 2, 4, …).

3. Can you make a machine which only accepts strings which begin and end with the same letter? To be definite: the alphabet should be {a,b}, and each accepted string should be at least two letters long, either starting and ending with an ‘a’, or starting and ending with a ‘b’. (Hint: split off at the beginning on a and b to two different states; then make little “sub-machines” that are the opposite of each other, a/b-wise).

4. How can you modify a machine so that it accepts the **opposite** language, i.e., every string (over the same alphabet) that the modified machine accepts, the original one rejected, and vice versa? (Hint: there is a quick trick to do this that was mentioned in lecture!)