COSC 4200
Computability & Complexity

Name: Jacob Tuttle

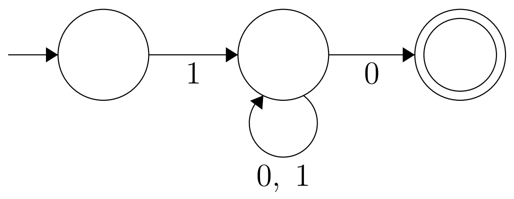
Homework 02

February 20, 2020

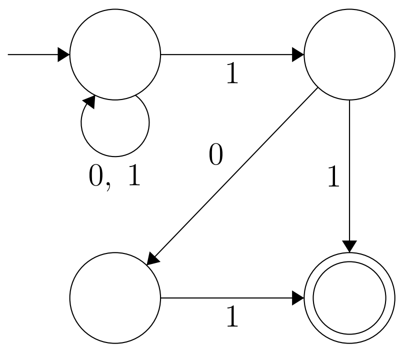
1. (a) A DFA can be constructed through product construction to show that $A - B$ is regular. Let Q be all states (a_i, b_j) where $(a_i, b_j) \in A \times B$. Then, let δ^* be the extended transition function created in the construction. Finally, include the pair (a_i, b_j) in F if $a_i \in F_A$ and $b_j \in F_B$. In this way, we define a DFA that accepts $A - B$ demonstrating it is regular.

   (b) Using the above demonstration that $A - B$ is regular, let us next construct the DFA corresponding to $B - A$. Then, since regular languages are closed under union, we know that $A \Delta B$ is regular.

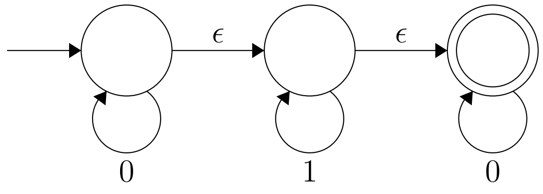
2. (a) This NFA will check that a 1 is present first, a 0 is present in the last position, and then allow anything else to be read in between that.



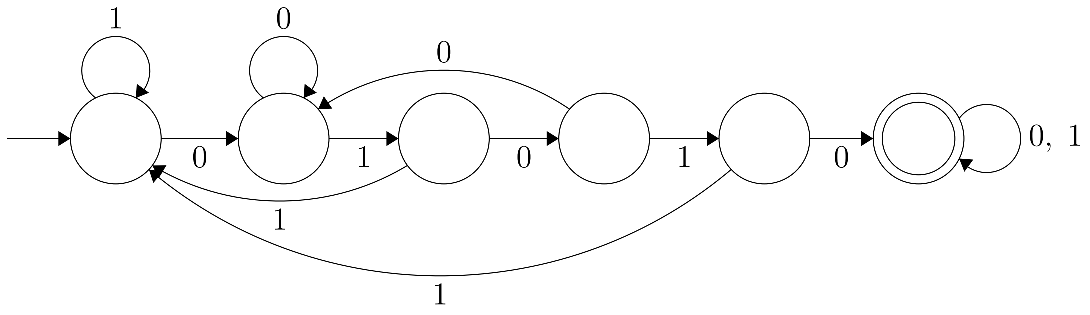
   (b) This will allow any beginning to the string, and as soon as a 101 or 11 is present at the end of the string, the NFA will escape to the success state.



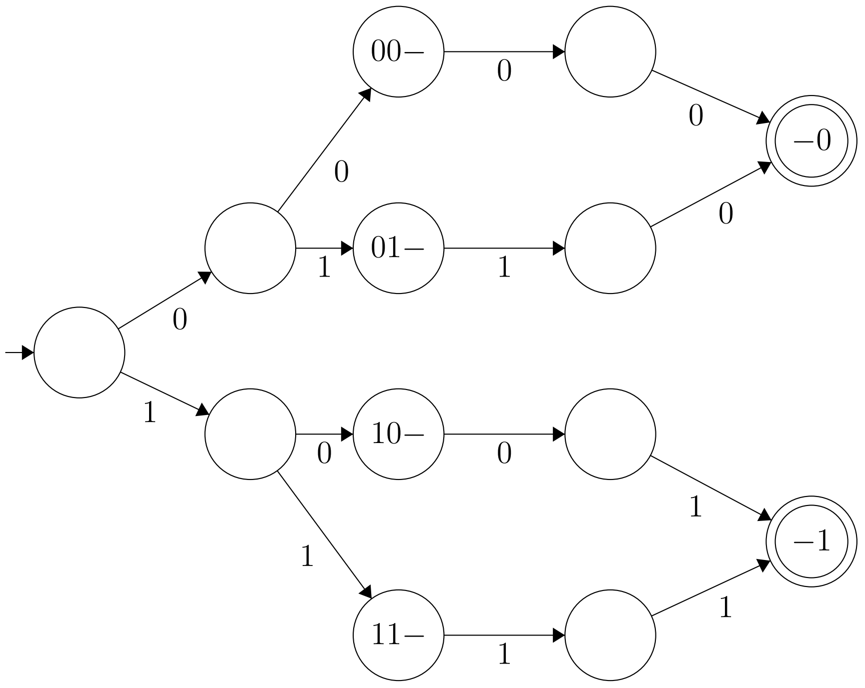
   (c) This NFA provides free paths for constructions where any of $i, j, k \leq 1$. But, an unacceptable construction would be unnable to make it to the final state because characters would be unread upon reaching the final accessible state and it would then fail.



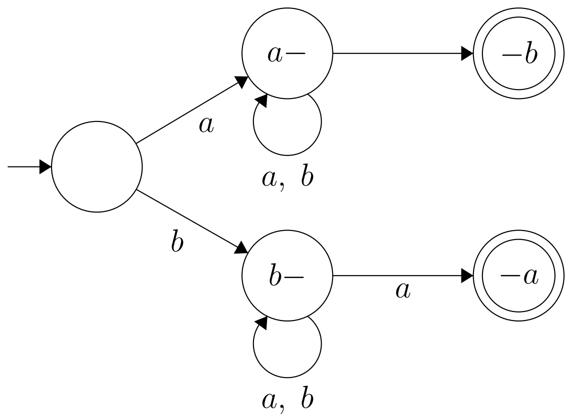
(d) This NFA will track through states used to determine whether the substring '01010' If the substring is present, then it will exit to a catch-all success state. Otherwise, it will continue moving through failure states.



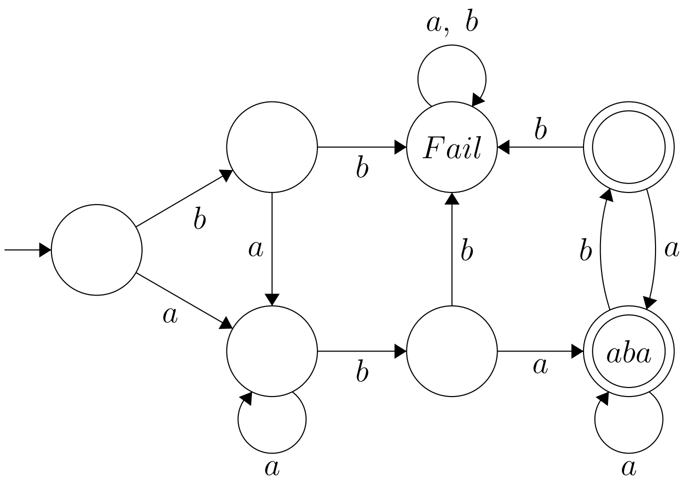
(e) This NFA moves to one of four branches to keep track of the first two characters that are read into the NFA. Then, the NFA will continue looping until it's able to follow a path out to an acceptance state that is the reverse of the first two characters. This NFA representation has 2 less states and 12 less edges.



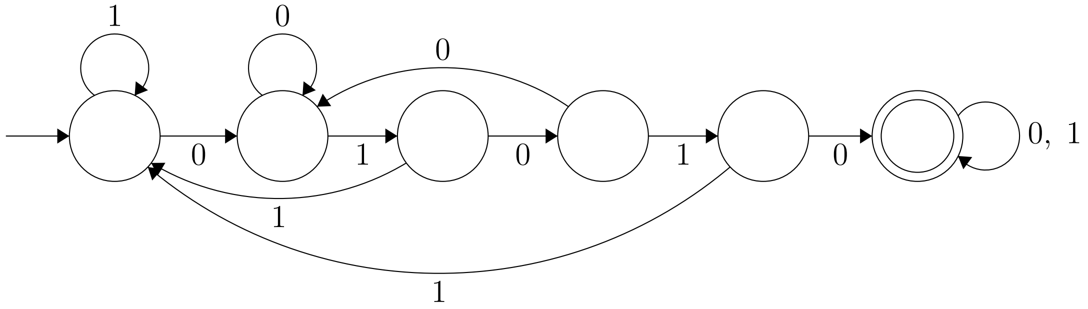
(f) This NFA will jump to one of two branches that keeps track of the first character read. Then, if the last character is different than the first character read, the NFA will jump to an acceptance state.



3. This NFA will drop into a fail state at any point where two bs are present in a row and moves to the right hand side where acceptance states are present after an aba has been read.



4. The NFA provided in 2d is already a DFA. Every state has a 0 and 1 transition coming from it, and there are no ϵ transitions in the machine.



5. (a) This language descripes all even length strings.
   (b) This language describes all strings of length ≤ 3.
   (c) This language describes all strings with substring '111' followed somewhere by substring '000.'

(d) This language describes all striongs that start with any number of 0s followed by either 0s or '100' repeatedly.

(e) This language describes all strings made of 0s and 1s.

6. (a) $[(a \cup b)^{\star} b (a \cup b)^{\star} b (a \cup b)^{\star} b (a \cup b)^{\star}]$

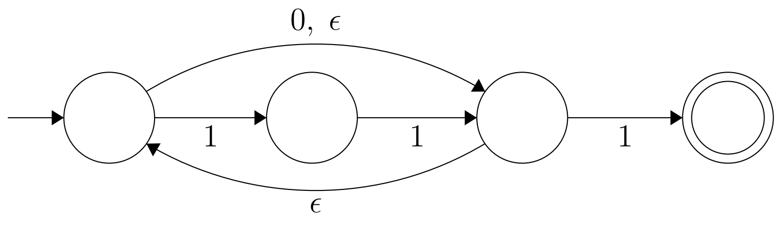
   (b) $[(a \cup b)[(a \cup b) b (a \cup b)]^{\star}[(a \cup b) \cup \epsilon] \cup \epsilon]$

   (c) $[b^{\star}(ab^{\star}(ab^{\star} \cup \epsilon) \cup \epsilon) \cup \epsilon]$

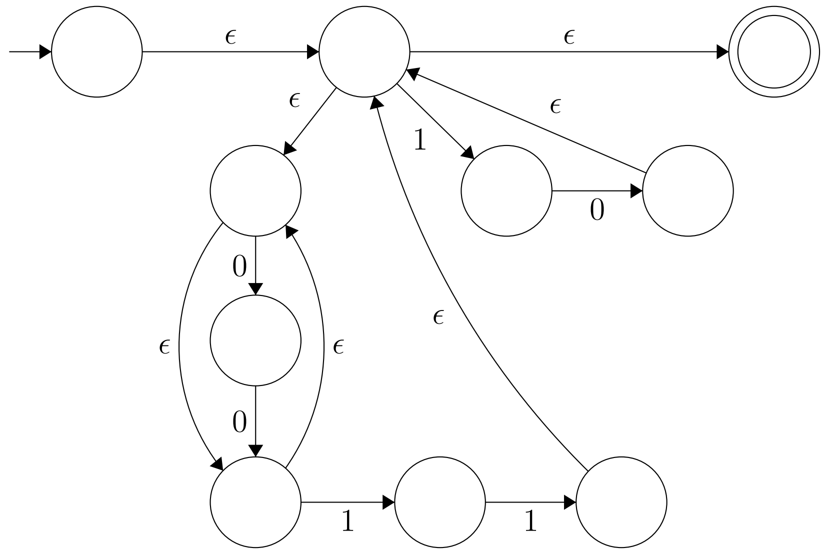
   (d) $[(aaaaaa)^{\star}(aaaa \cup [aaa \cup (aa \cup \epsilon)])]$

   (e) $[b^{\star}(ab^{\star}[ab^{\star}a(a \cup b)^{\star} \cup \epsilon] \cup \epsilon)]$

7. (a) This NFA represents the regular expression $[(0 \cup 11)^{\star}(1 \cup 01)]$



   (b) This NFA represents the regular expression $[(00)^{\star}11 \cup 10]^{\star}$



8. Constructed through reduction on a DFA representing the states, the regular expression is $[1([0010^{\star} \cup 1][101^{\star}0]^{\star}0)^{\star}([0010^{\star} \cup 1][101^{\star}0]^{\star}11 \cup 01)] \cup 0^{\star}$