COSC 4200 Name: Jacob Tuttle
Computability & Complexity

Homework 03

February 27, 2020

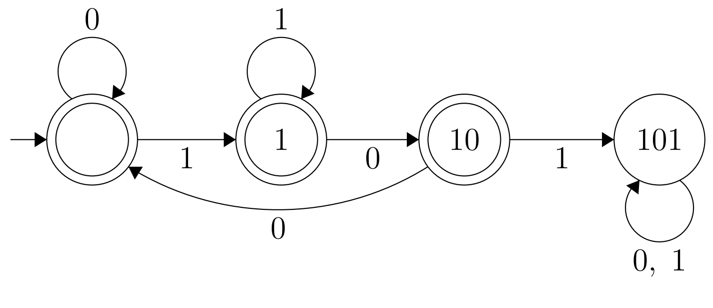
1. (a) Perform a product construction of A and B where the only accepting states are ones in which both A and B accept. Then, perform a product construction of that new DFA with C. Accepting states are states in which $A \cap B$ or C are accepting. The DFA will have $n_a n_b n_c$ states.

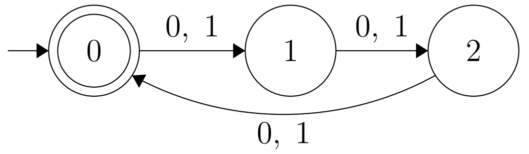
   (b) Perform a product construction on A and B. The accepting states will be all states where either A or B accepts. Then perform a cline closure by taking the union of all possible $(A \cup B)^k$ for all $k \leq \infty$. This effectively tacks on every possible combination of strings in $A \cup B$ into the closure. There are an indeterminate amount of states.

   (c) Perform a concatination of A and B by establishing free moves from accepting states in A to the first state in B. Convert that NFA to a DFA. Then, use product construction on the resulting DFA with C where accepting states are ones in which $A \cdot B$ and C are in accepting states. The DFA will have $(n_a + n_b) * n_c$ states.

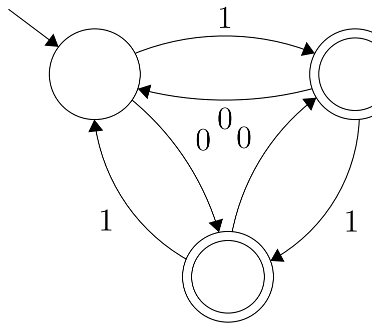
   (d) Perform a cline closure as described in part (b) on A. Then perform a product construction on the resulting $A*$ and B. Accepting states are all states in which both $A*$ and B are accepting. Then take all states that aren't accepting states and make them accept while taking all previously accepting states and making them reject to complete the complement. Finally, perform the concatination operation as described in (c) on the new DFA and C.

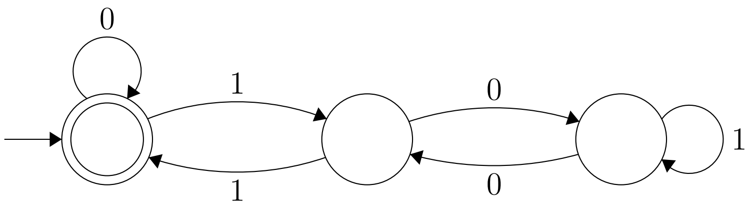
2. (a) This DFA can be formed by taking the union of two DFAs: one that fails if the substring 101 is present (top) and one that succeeds when the length of x is a multiple of 3 (bottom).



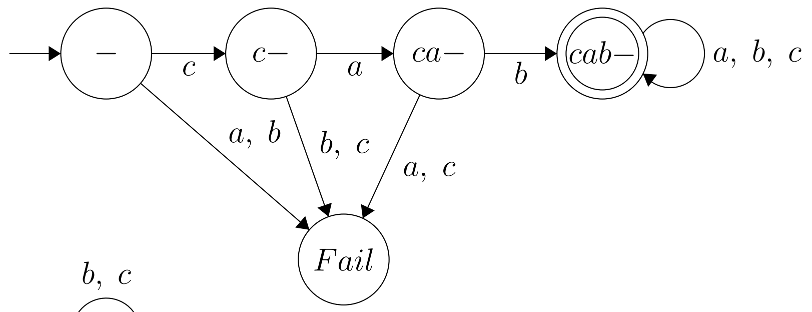


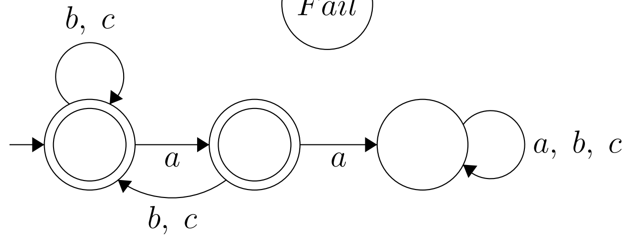
(b) This DFA is most easily represented by the intersection of two sub-DFAs, presented below. One counts the difference between the number of 0s and the number of 1s (top) and one counts the binary value of x mod 3 (bottom).

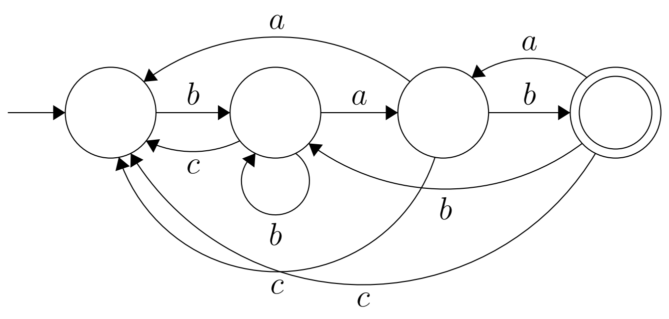




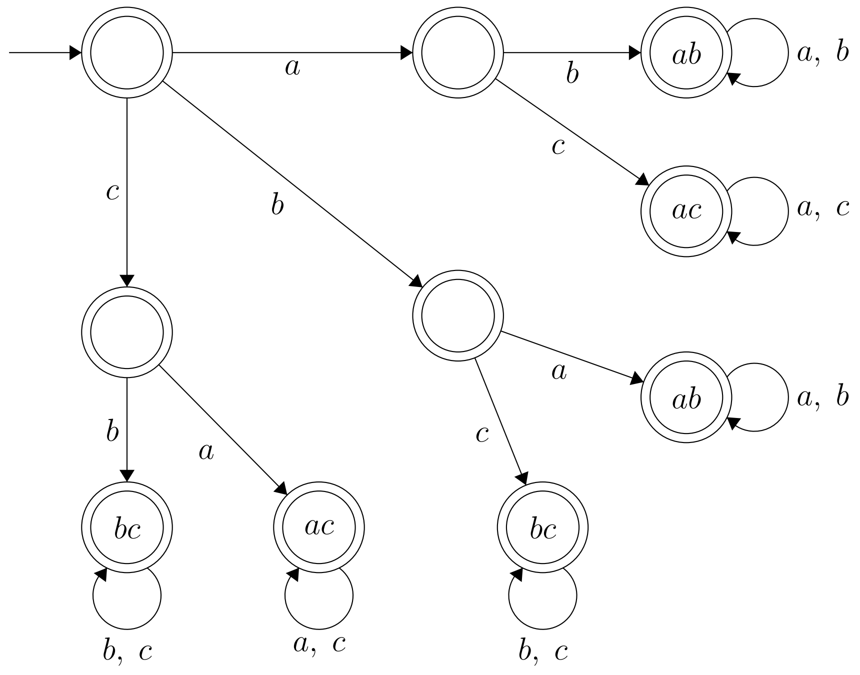
(c) This DFA can be formed by taking the intersection of 3 sub-DFAs. One checks if a string begins with 'cab,' the next checks for strings that don't contain the substring 'aa,' anf the final one checks for a string that ends with 'bab.'



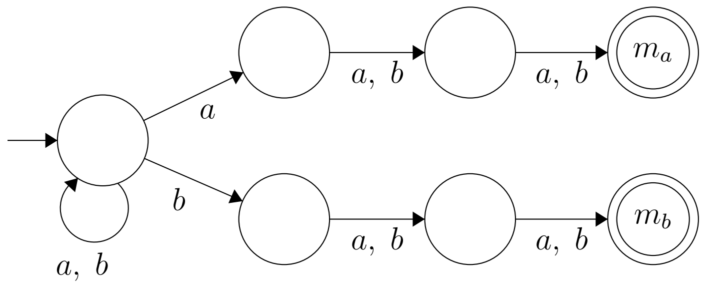


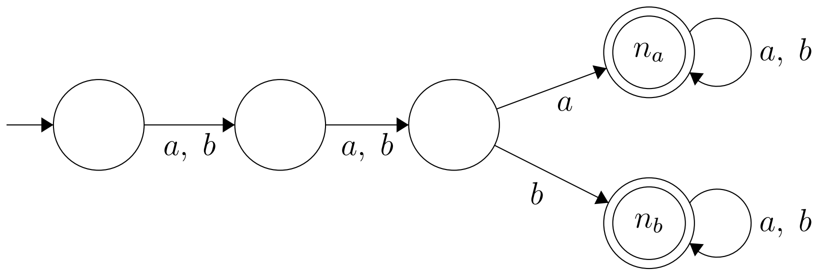


(d) The following NFA will check to see that there are only two characters present. As soon as a third is present in the string, there is nowhere to proceed and the machine fails. (The machine is presented in an unreduced form to make it easier to understand at a glance without having to decipher criss crossed lines. The accepting states labeled with the same letters could be combined into 1 state each.)

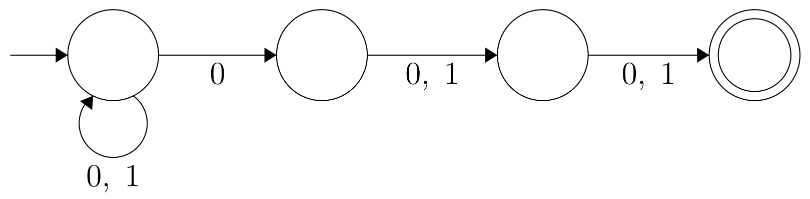


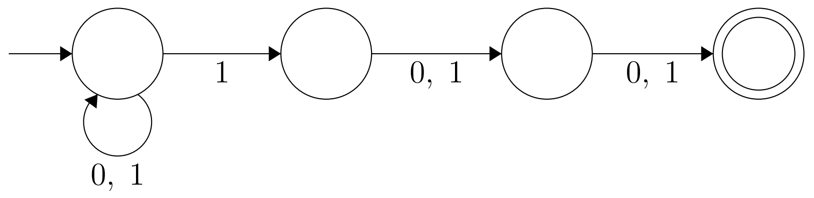
(e) This NFA can be produced using careful product construction of two NFAs: one that tracks the third to last bit (top) and one that tracks the third bit (bottom). Let the top NFA be M and the bottom NFA be N. The accepting states of $M \times N$ is given as $F = \{m_a n_a, m_b n_b\}$ indicating that the 3rd first and 3rd last bits are the same.

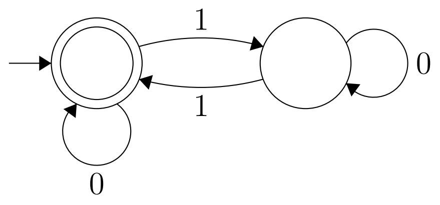


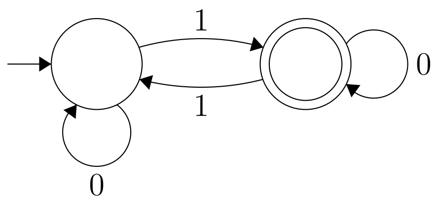


(f) This NFA is created through applciation of closure properties on the sub NFAs below. First let us label them from top to bottom as $B_0, B_1, P_0,$ and P_1 respectively. They represent the third to last bit being 0, the third from last bit being 1, the parity being 0, and the parity being 1 respectively as well. The NFA combining all of these features is given as $(P_0 \cap B_0) \cup (P_1 \cap B_1)$.

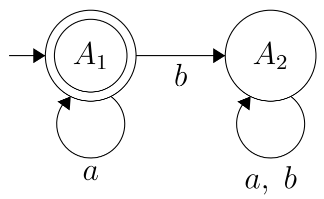


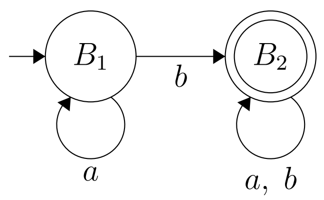






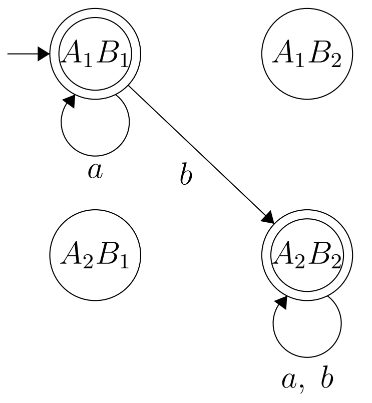
3. Let L_A be the language accepted by A where $L_A = \{x \in \{a, b\}^* | 'b' \in x\}$. Let L_B be the language accepted by B where $L_B = \{x \in \{a, b\}^* | 'b' \notin x\}$. They can be seen below:

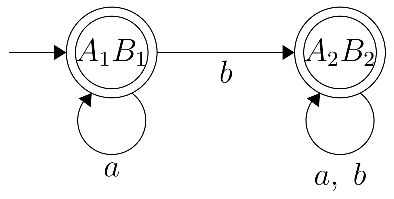




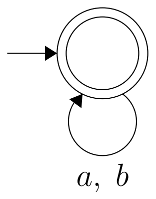
Next, let $L_{A\times B}$ be the language accepted by a DFA that is created with product construction from $A \cup B$. This DFA looks like this and accepts $\{x \in \{a, b\}^* | 'b' \in x$ or

$'b' \notin x\}$. On the left is the direct product construction DFA, and on the right is the same DFA with unreachable states removed.





This DFA is not a minimal DFA as demonstrated by the below DFA. It accepts all of the same strings and has only 1 state as opposed to 2 states.



4. Suppose D is regular. Let p be the pumping constant for D. Let $w = 0^p1^{2p}$. Then w can be broken up into substrings xyz where $|xy| \le p$ and $0 \le |y|$ such that xy^iz exists for all i.

   Then we know that $x = 0^k, y = 0^l$, and $z = 0^{p-k-l}1^{2p}$ where $k \ge 0, l > 0$.

   Let $i = 2$. Then:

$$\begin{aligned} xy^2z &= 0^k0^{2l}0^{p-k-l}1^{2p} \\ &= 0^{p+l}1^{2p} \in D \end{aligned}$$

   But since $l \ge 0, p + l \ne p$ Thus, there are more 0s than D is able to accept. Since this is a contradiction to our assumption that D is regular, we know that it must be nonregular.