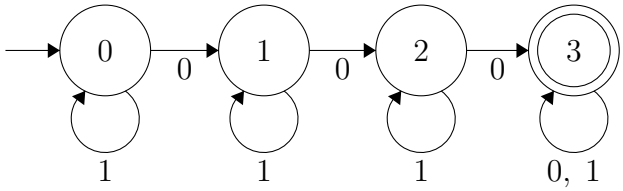
COSC 4200
Computability & Complexity

Name: Jacob Tuttle

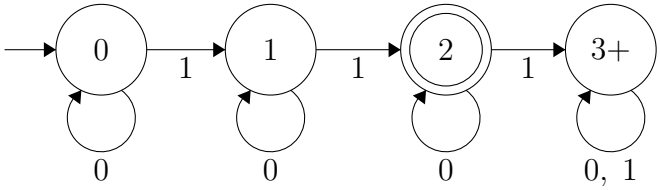
Homework 01

February 10, 2020

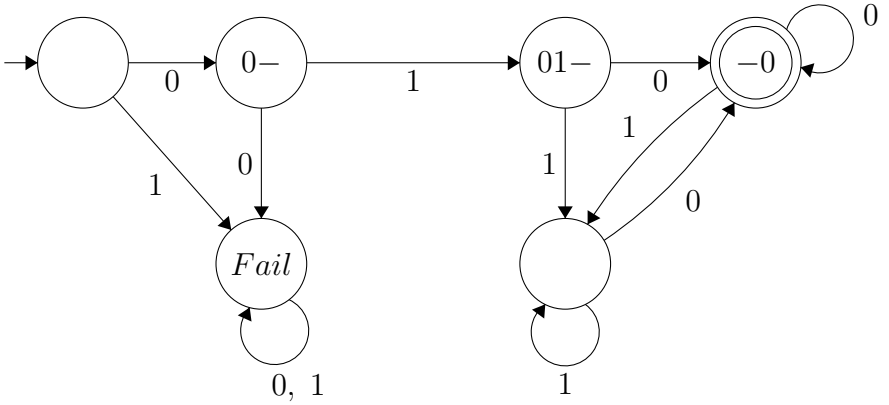
1. (a) This DFA counts the number of 0s until the acceptance condition is met. At that point, any further input will always result in an accepted state, so it simply reads through the remaining inputs.



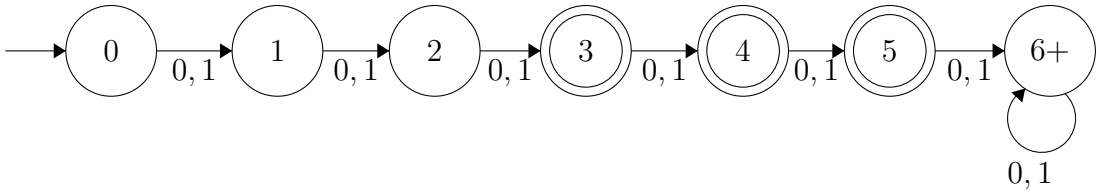
(b) This DFA counts the number of 1s present within the input word and will accept the word when there are exactly two 1s within the word. After 3 1s are read, it simply reads through the rest of the word and remains in a failure state.



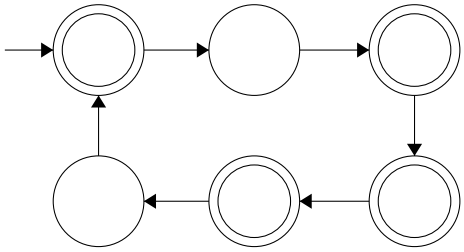
(c) First, the DFA will ensure that the 01 prefix is within the string. If it reads a different set of first two characters, it will move to, and terminate in, the state labelled ”Fail.” Otherwise, it will read through the rest of the word, switching between the acceptance state and a non-accepting state until it reaches the end of the word and has checked if 0 is the last character in the word.



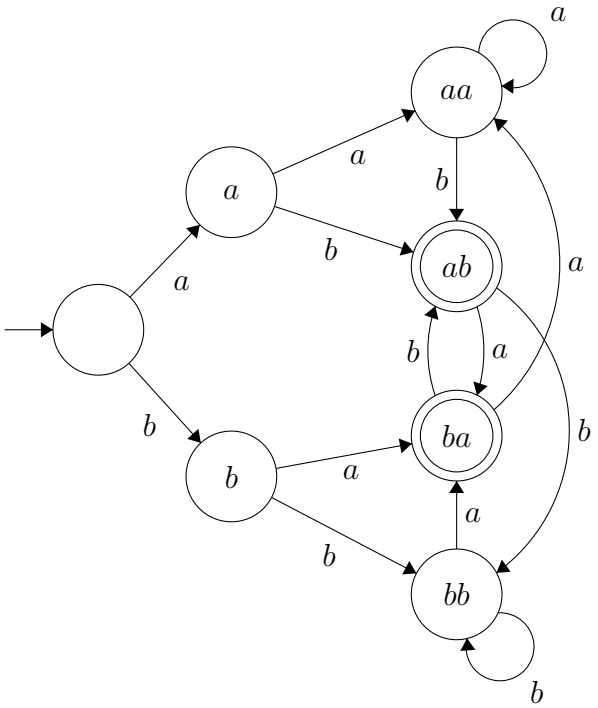
(d) This DFA will progress through states labeled with the length of the word until it reaches the end of the word. If the length of the word is between 3 and 5 inclusively, it will end in an acceptance state, but if the word length is six or more characters, it will simply remain in the 6+ state until the word has been fully read.



(e) This DFA works by constructing a circuit that is the length of the least common multiple of 2 and 3, namely 6, states. The states, in order from the input state represent the length being divisible by: both 2 and 3, neither, only 2, only 3, only 2, neither, and both 2 and 3. Since this pattern repeats as length grows to infinity, however long the word is will be accurately counted and then properly divided by this DFA. Also note, all state transitions will transition on an input of 'a' but are not marked as such since only one input character is possible in the given alphabet.

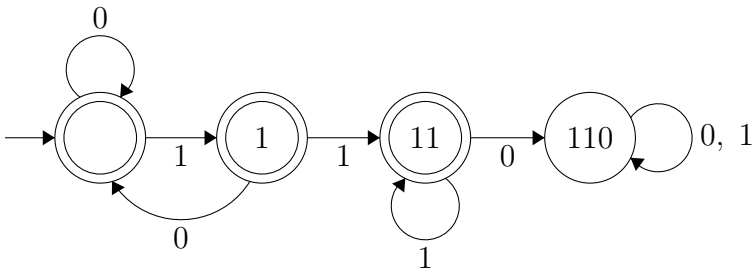


(f) This DFA keeps track of the last two read characters. If they are different, and the entire word has been read, then the word ends with either ba or ab. Otherwise, the DFA will finish in a non-accepting state.



(g) This DFA keeps track of substrings that begin to look like the non-accepting substring 110. If That substring is recognized within the DFA, it moves to an

always non-accepting state. Otherwise, regardless of which state the DFA is in, it will accept the word since the substring does not appear within it.



(h) This DFA is best represented by a union of the three DFAs that make it up. These will check that there is at least 1 a, no more than 3 bs, and exactly 1 c. We will define the DFA that combines these three smaller DFAs mathematically. $Q = \{a_i b_j c_k | 0 \leq i \leq 1, 0 \leq j \leq 4, 0 \leq k \leq 2\}$; $\Sigma = \{a, b, c\}$; δ will be defined in parts below; $a_0 b_0 c_0$ is the initial state; and $F = \{a_1 b_0 c_1, a_1 b_1 c_1, a_1 b_2 c_1, a_1 b_3 c_1\}$.

Let δ' be the function that describes the joint DFA for determining the number of bs and cs. We will define δ' for all $0 \leq i \leq 3$ as:

$$\begin{array}{lll} \delta'(b_i c_0, a) = b_i c_0 & \delta'(b_i c_1, a) = b_i c_1 & \delta'(b_i c_2, a) = b_i c_2 \\ \delta'(b_i c_0, b) = b_{i+1} c_0 & \delta'(b_i c_1, b) = b_{i+1} c_1 & \delta'(b_i c_2, b) = b_{i+1} c_2 \\ \delta'(b_i c_0, c) = b_i c_1 & \delta'(b_i c_1, c) = b_i c_2 & \delta'(b_i c_2, c) = b_i c_2 \end{array}$$

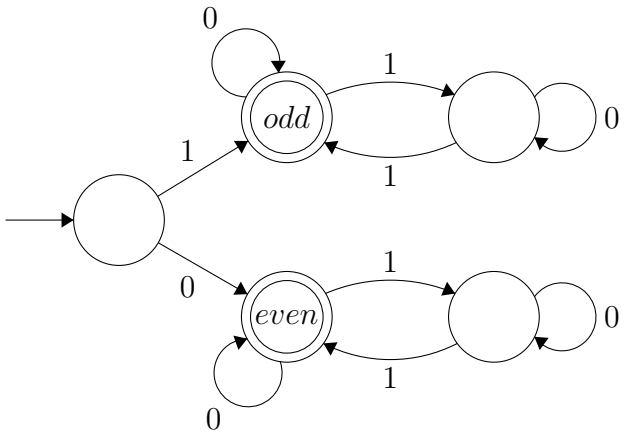
$$\begin{array}{lll} \delta'(b_4 c_0, a) = b_4 c_0 & \delta'(b_4 c_1, a) = b_4 c_1 & \delta'(b_4 c_2, a) = b_4 c_2 \\ \delta'(b_4 c_0, b) = b_4 c_0 & \delta'(b_4 c_1, b) = b_4 c_1 & \delta'(b_4 c_2, b) = b_4 c_2 \\ \delta'(b_4 c_0, c) = b_4 c_1 & \delta'(b_4 c_1, c) = b_4 c_2 & \delta'(b_4 c_2, c) = b_4 c_2 \end{array}$$

From this definition, we can then combine it with the DFA to count as by piecing togethether this definition of δ'' using a state $b_i c_j$ as x

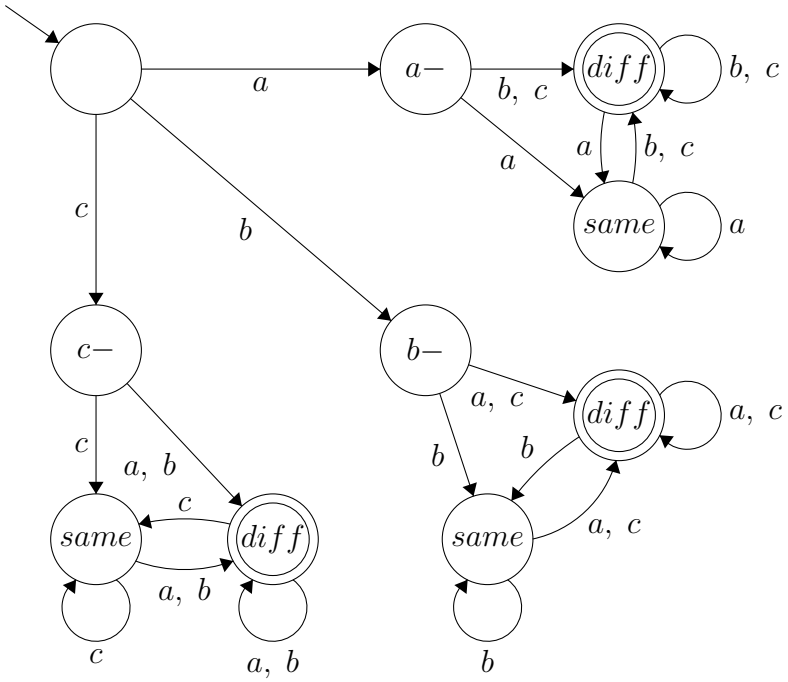
$$\begin{array}{ll} \delta''(a_0 x, a) = a_1 \delta'(x, a) & \delta''(a_1 x, a) = a_1 \delta'(x, a) \\ \delta''(a_0 x, b) = a_0 \delta'(x, b) & \delta''(a_1 x, b) = a_1 \delta'(x, b) \\ \delta''(a_0 x, c) = a_0 \delta'(x, c) & \delta''(a_1 x, c) = a_1 \delta'(x, c) \end{array}$$

For instance, $\delta''(a_1 b_0 c_0, b) = a_1 \delta'(b_0 c_0, b) = a_1 b_1 c_0$. This final result of evaluating δ'' is the transition function δ. Thus, we have a DFA that will end up in an accepting state if there is at least 1 a (represented by the partial state a_1), there are at most 3 bs (represented by the inclusions of partial states b_0 through b_3), and there is exactly 1 c (represented by partial state c_1). All other states are excluded from the DFA's set of accepting states since at least one of the three criteria for acceptance would not have been met.

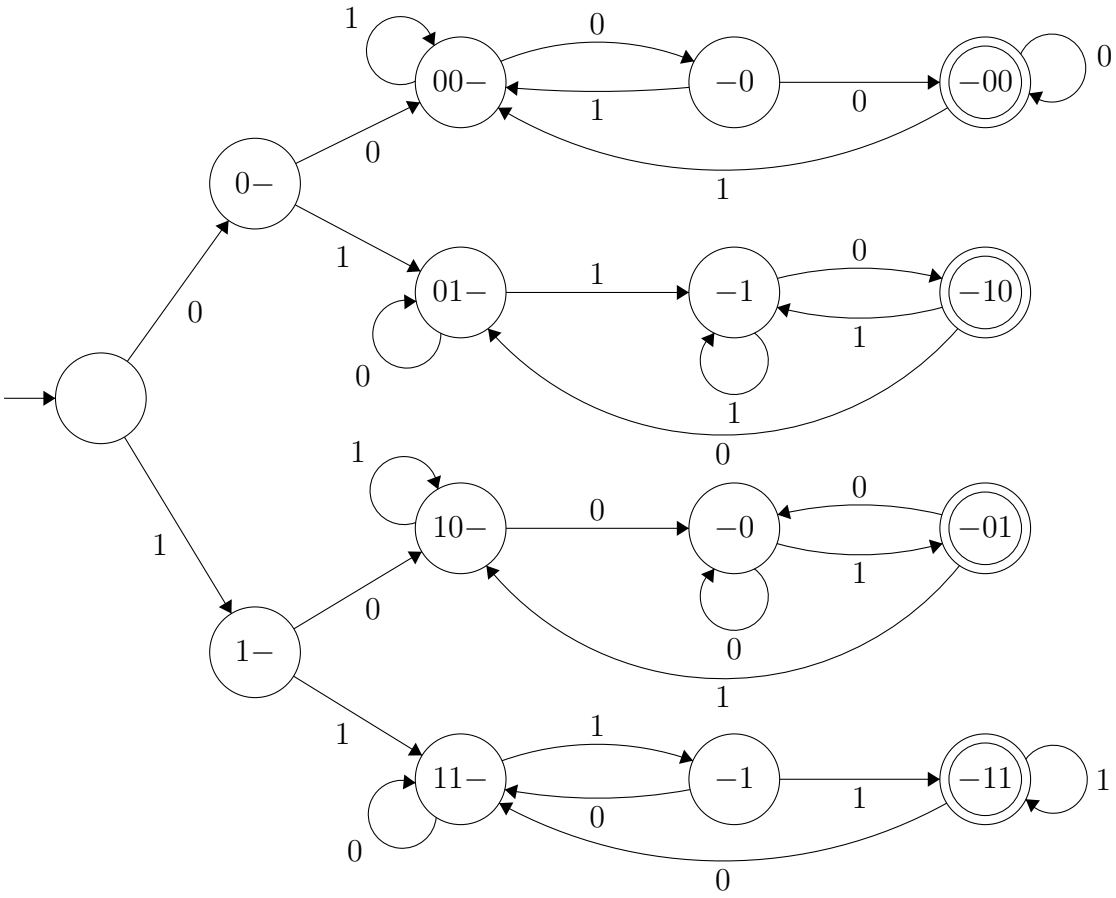
(i) Based on the input character, this DFA will branch into one of two sub-DFAs that measures the parity. They both alternate whenever a 1 is read to oscilate between matching parity and mismatched parity. If the parity matches the first bit, then the DFA will accept.



(j) This DFA will branch into one of three different ”paths” to keep track of what the first symbol is. Then, as subsequent letters are read, the DFA will move to an accepting state if the next characters are different than the first one, and it will move to a non-accepting state if the first symbol matches the last read symbol.



2. This DFA will read in the first two characters of the word, w and then continue reading characters. Each of the "arms" of the DFA will look for the substring w^R, and if that substring is the end of the input word, accept the input. Otherwise, it will fail since the input doesn't match the construction rules for the language.



3. This DFA functions on the same principle as the binary DFA demonstrated in lecture. Using the inductive definition of a trinary system, the DFA will move to the state represented by 3 times the value of the current state + the input value modulo 5. The transition function δ is $\delta(q, b) = 3 * q_{val} + b$ where q_{val} is the value represented by each state.