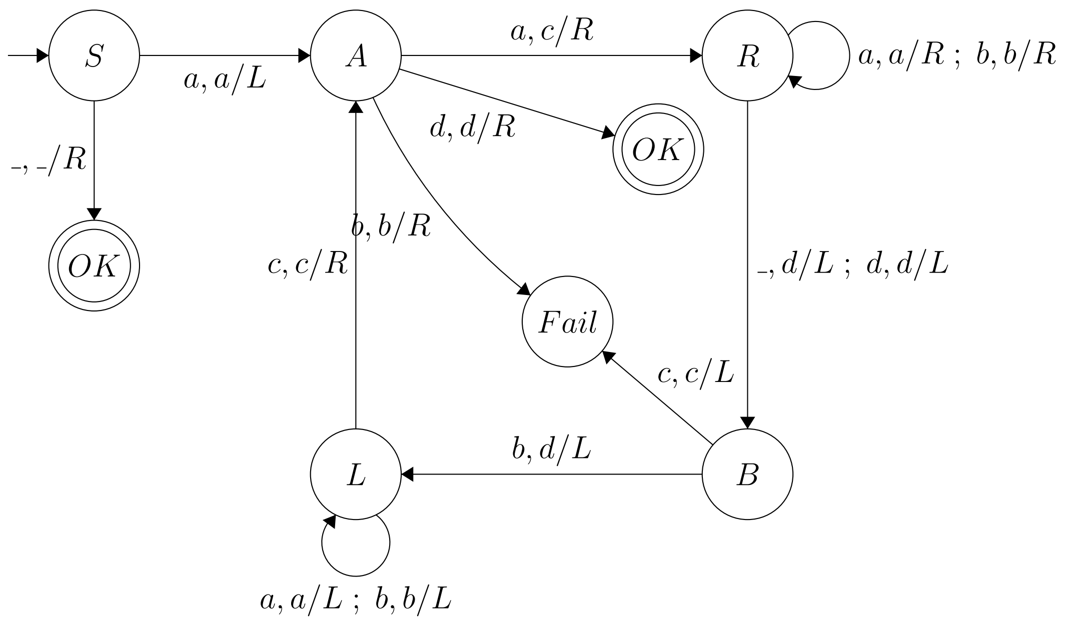
COSC 4200 Name: Jacob Tuttle
Computability & Complexity

Homework 05

April 16, 2020

1. Where notation a,b/L or R indicates reading an a in the current cell, writing a b to the current cell and then moving either left (L) or right (R), the following Turing Machine state-transition diagram will accept strings where a string of as is followed by the same number of bs. The TM works by first checking if the input is the empty string (n = 0). If so, it moves to the OK state, which is the only accepting state (it is rendered twice in this diagram for clarity, but both states indicated OK refer to a single state.) Then, since our TM will remain at the first position if it tries to move left from there, we maintain the value in the first spot and move left, which keeps the machine at the start of the input string. Then, it replaces the first character in the string with an indicator that the number has been read and begins moving right. The machine moves right until it either reaches an empty cell (i.e. the end of the input) or the terminal character for bs, then the head moves left one character. It then replaces the b with a right hand terminal character and then moves left until it reaches a terminal on the left. Then it moves right once. At this point, the first and last character have both been dropped from the string. This process can be repeated to continue peeling off as many pairs as exist. If there are more as than bs, when the TM is in state B, it will read a 'c' instead of a 'b' and so transition to a Fail state. If there are more bs than as, when the TM is in state A, it will read a 'b' instad of an 'a' and so it will transition to a fail state. Finally, if there are the same number of 'a's and 'b's, state A will read a 'd' and transition to an accepting state.



2. (a) Designate two new alternate languages that we will use to designate the first and second half of the string. Let a,b be the left language L and c, d be the right language R.
   (b) Remember the character in the current cell, and write a terminal character '#'.
   (c) Move right one cell and remember the character in the cell while writing the character from the previous cell.
   (d) Repeat steps (b)-(d) until the entire string has been shifted one space to the right and the TM head is sitting at the first empty cell after the input string. Write the terminal character #.
   (e) Return to the beginning of the string by moving left until you read the terminal character and then moving right once.
   (f) aaaRead the character in the current cell and replace it with the corresponding symbol from L: a if the TM read a 0 and b if it read a 1.
   (g) Move right until you reach a terminal character or a character in R. Then move left once.
   (h) Replace the character in the current cell with the corresponding symbol from R: c if the TM read a 0 and d if it read a 1.
   (i) Move left until you reach a symbol in L. Then move right once.
   (j) If the current character isn't in R, repeat steps (f)-(j).
   (k) Move left until you reach #, then move right once. Remember the current character, replacing it with an ␣.
   (l) Move right until you reach a symbol in R.
   (m) If the current symbol corresponds to the same symbol encoded in memory by language L, replace the current cell with an ␣, if they don't match, reject.
   (n) Move left until you reach a #.
   (o) Move right until you reach a character in L. If you read #, accept.
   (p) Repeat steps (k)-(p).

3. Let P be the language $\{< N, R > \mid N$ is an NFA & R is a regular expression where $L(N) = L(R)\}$

   Then, let the following Turing Machine, T decide P:

   $T =$ "On input $< N, R >$ where N is an NFA and R is a regular expression:

   (a) Use the algorithms from class to convert the regular expression R to NFA R_{NFA}.
   (b) Use subset construction to convert NFA N to DFA N_{DFA} and NFA R_{NFA} to DFA R_{DFA}.
   (c) Run the TM EQ_{DFA} from lecture with inputs N_{DFA} and R_{DFA}.
   (d) If EQ_{DFA} accepts, accept. If EQ_{DFA} rejects, reject.

4. We can show that All_{DFA} is decidable by designing a decider for it. Let D be this decider and define it as such: $D =$ "On input M where M is a DFA:

   (a) Using algorithm T from the emptiness problem in lecture, run algorithm T on M.
   (b) if $L(M) = \emptyset$, reject. Otherwise, accept.

5. Let us assume that A_{101} is decidable, and T_{101} is the turing machine that decides A_{101}. T_{101} is the machine that simulates A_{101} and accepts if A_{101} accepts and rejects otherwise. Then, we will run the Diagonalize function, D on T_{101}. $D < T_{101} >$ accepts if $< T_{101} >$ rejects and vice versa. But, since we can run $D < D < M >>$ on itself, we know that this will accept when it rejects and reject when it accepts, which is a contradiction, and therefore A_{101} is undecideable.