Anthony Morrell
CSIS 616

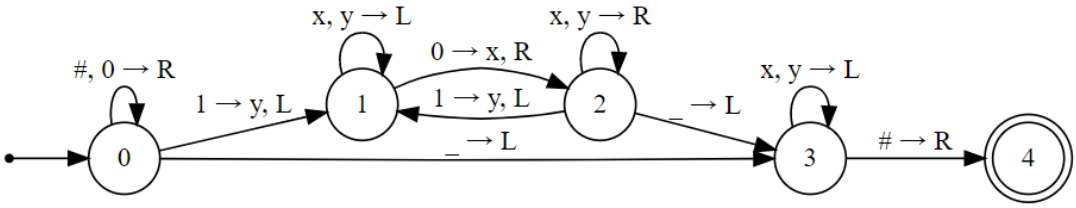
Chapter 3 Homework

Problem A

1. The machine expects a string of 0's and 1's beginning with a #. It starts by scanning right for the first 1 and replaces it with an Y. It then scans left for the last 0 and replaces it with an X. It repeats these two scans until it reaches the end of the input. At this point, it scans back to the other side to make sure there are no remaining numbers. If there aren't, it accepts. If at any point there is no defined transition, it rejects.

2. $M = (Q = \{q_0..q_4\}, \Sigma = \{0,1,\#\}, \Gamma = \{0,1,\#,x,y,\_\}, \delta, q_0 = q_0, q_{accept} = q_4)$
   $\delta =$

	0	1	#	x	y	_
q0	q0, 0, R	q1, y, L	q0, #, R			
q1	q2, x, R			q1, x, L	q1, y, L	
q2		q1, y, L		q2, x, R	q2, y, R	q3, _, R
q3			q4, #, R	q3, x, L	q3, y, L	
q4						

3. 

4. q0 #0011
   # q0 0011
   #0 q0 011
   #00 q0 11
   #0 q1 0y1
   #0x q2 y1
   #0xy q2 1
   #0x q1 yy
   #0 q1 xyy
   # q1 0xyy
   #x q2 xyy
   #xx q2 yy
   #xxy q2 y
   #xxyy q2
   #xxy q3 y
   #xx q3 yy
   #x q3 xyy
   # q3 xxyy
   q3 #xxyy
   # q4 xxyy -- ACCEPT

5. q0 #0010
   # q0 0010
   #0 q0 010
   #00 q0 10
   #0 q1 0y0
   #0x q2 y0
   #0xy q2 0 -- REJECT

Anthony Morrell
CSIS 616

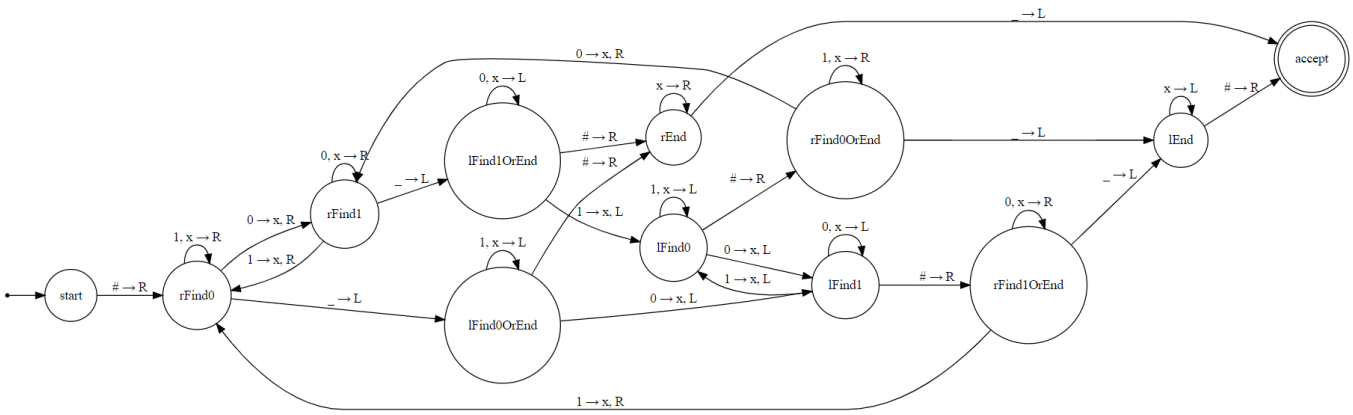
Problem B

1. The machine expects a string of 0's and 1's beginning with a #. It starts by scanning right for either 0's or 1's (beginning with 0's). After each find, it replaces the number with an X and continues scanning for the other number. When it reaches the end, it turns around and continues the process in the other direction, but in a special variant state. If it finds the number it's looking for while in this variant state, it switches back to the regular state and continues scanning. However, if it reaches the end in this state, this means there are no more occurrences of at least one of the numbers on the tape. It performs one final pass to check for any remaining numbers on the tape. If it finds none, it accepts. If at any point there is no defined transition, it rejects.
2. $M = ($
   $Q = \{start, rFind0, rFind1, lFind0, lFind1, rFind0OrEnd, rFind1OrEnd, lFind0OrEnd, lFind1OrEnd, rEnd, lEnd, accept\}$,
   $\Sigma = \{0, 1, \#\}, \Gamma = \{0,1, \#, x, \_\}, \delta, q_0 = start, q_{accept} = accept$
)
   $\delta =$

	0	1	x	#	_
start				rFind0, #, R	
rFind0	rFind1, x, R	rFind0, 1, R	rFind0, x, R		lFind1OrEnd, _, L
rFind1	rFind1, 0, R	rFind0, x, R	rFind1, x, R		lFind0OrEnd, _, L
lFind0	lFind1, x, L	lFind0, 1, L	lFind0, x, L	rFind1OrEnd, #, R	
lFind1	lFind1, 0, L	lFind0, x, L	lFind1, x, L	rFind0OrEnd, #, R	
rFind0OrEnd	rFind1, x, R	rFind0OrEnd, 1, R	rFind0OrEnd, x, R		lEnd, _, L
rFind1OrEnd	rFind1OrEnd, 0, R	rFind1, x, R	rFind1OrEnd, x, R		lEnd, _, L
lFind0OrEnd	lFind1, x, L	lFind0OrEnd, 1, L	lFind0OrEnd, x, L	rEnd, #, R	
lFind1OrEnd	lFind1OrEnd, 0, L	lFind1, x, L	lFind1OrEnd, x, L	rEnd, #, R	
rEnd			rEnd, x, R		accept, _, L
lEnd			lEnd, x, L	accept, #, R	
accept					

3. (state diagram on next page)

Anthony Morrell
CSIS 616



Problem C

You can use the empty tape to the right of the input as a stack. The machine could place the stack start symbol after the end of the input to separate the “input tape” from the “stack tape”. To push a value onto the stack, the machine could replace the first blank with the pushed value. To pop a value off the stack, the machine could scan for the first blank and then “delete” (replace with a blank) the value to its left. To keep its location in the input, the machine would have to replace each “visited” letter with a special symbol that is not part of the input alphabet.

Problem D

The machine expects a string of 0’s and 1’s beginning with a #. It starts by looking at the first letter, replacing it with an X, and then scanning right. It stops (non-deterministically) at the start of the second *w* and checks if it matches the first letter, rejecting if it doesn’t match, or replacing it with a Y if it does. Now that the start of each word is marked by an X or Y, the machine can jump back and forth between the next letter in each half, replacing each one with another X or Y and rejecting if it discovers a mismatch. When it runs out of letters in one half, it finally scans to the end of the other half to make sure it has also run out of letters, accepting if it has and rejecting if it hasn’t.