Tut-9

Name: P.V.SRIRAM

Roll No.: 1801CS37

Q1) RTP: Semi-Infinite Machines have same power as standard turing machines.

Proof:

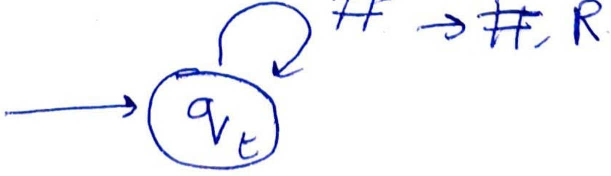
I) Standard turing machines simulate semi infinite machines.

Consider a standard turing machine and perform following modifications

(i) Insert '#' symbol on the left of input string

◇	◇	#	a	b	a	c	◇

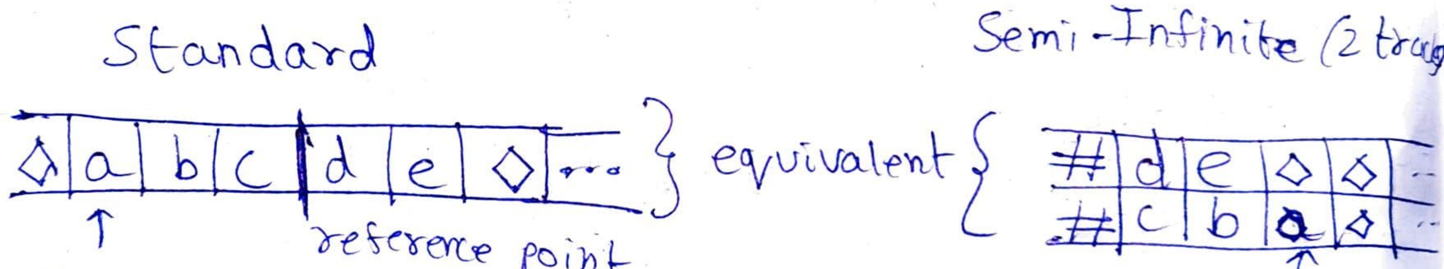
(ii) Add a self loop to each state of the standard turing machine.



Exclude states with no outgoing transition.

This will ensure the standard turing machine will not go beyond the #, so left infinity can never be reached. In this way semi-Infinite tape machine is simulated.

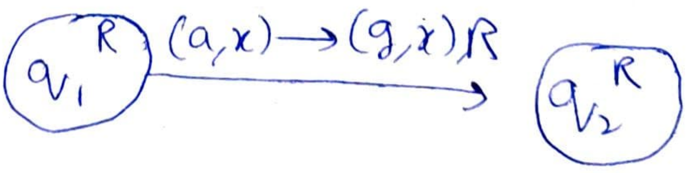
II) Semi-Infinite Machine simulates standard turing Machines since semi-Infinite machine has one infinity only, wheras standard turing machine has two, we use a semi-Infinite machine with 2 tracks for simulation.



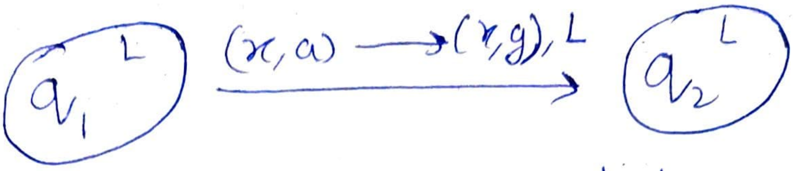
The head of semi-infinite machine will read both tracks.
Each state q_1 in standard turing machine has 2 counterparts q_1^L. q_1^R in semi-infinite machine.

With the following transitions included in the semi infinite machine, we ensure the functionality is same in both machines.

(1) Right Part



Left Part



For any tape symbol x.

Simulation:

Time 1

◇	a	b	c	d	e	◇
	↑					

Right Part

#	d	e	◇	◇
#	c	b	a	◇
			↑	

Left Part

Time 2

◇	g	b	c	d	e	◇
	↑					

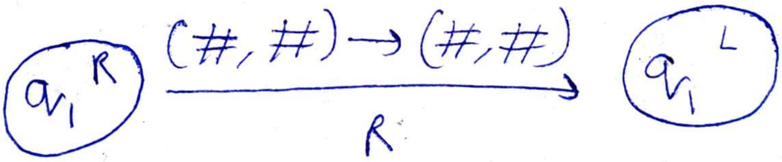
Right Part

#	d	e	◇	◇
#	c	b	g	◇
		↑		

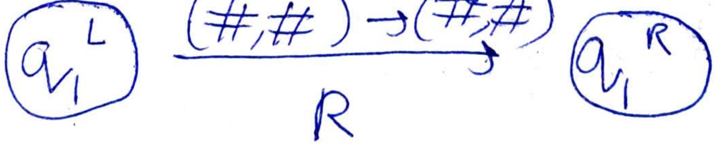
Left part

② At the border,

Right part



Left Part



Simulation

Time 1

Right	#	d	e	◇	◇
Left	#	c	b	g	◇

↑ q_1^L

Time 2

Right	#	d	e	◇	◇
Left	#	c	b	g	◇

↑ q_1^R (Now 'd' will be read so we moved to right)

In this way, we can prove that standard turing machine and semi-Infinite turing machine have same power. Hence proved

Q2)

$L(M) = \{ a^n : n \text{ is prime} \}$

M never changes its first tape, which holds the input a^n. It begins by rejecting if n is 0/1 and accepting if it is 2. Otherwise if its places two a's on its second tape. Then it puts each head at the left end of its string, then moving the tape-2 head back to the left end and repeating. If it reaches the end of the tape-1 string at the same time it reaches the end of the tape-2 string, it rejects.

Otherwise it adds a third a to the tape-2 string and marks off copies of the tape-2 string on the tape-1 string. If it finds they are equal length, it accepts. Otherwise it marks off copies and rejects if it finds the right ends of the two strings at same time. Otherwise it continues increasing the size of tape-2 string by one letter each time. It accepts if len(tape1) = len(tape2) and rejects if len(tape2) = multiple of len(tape 2)

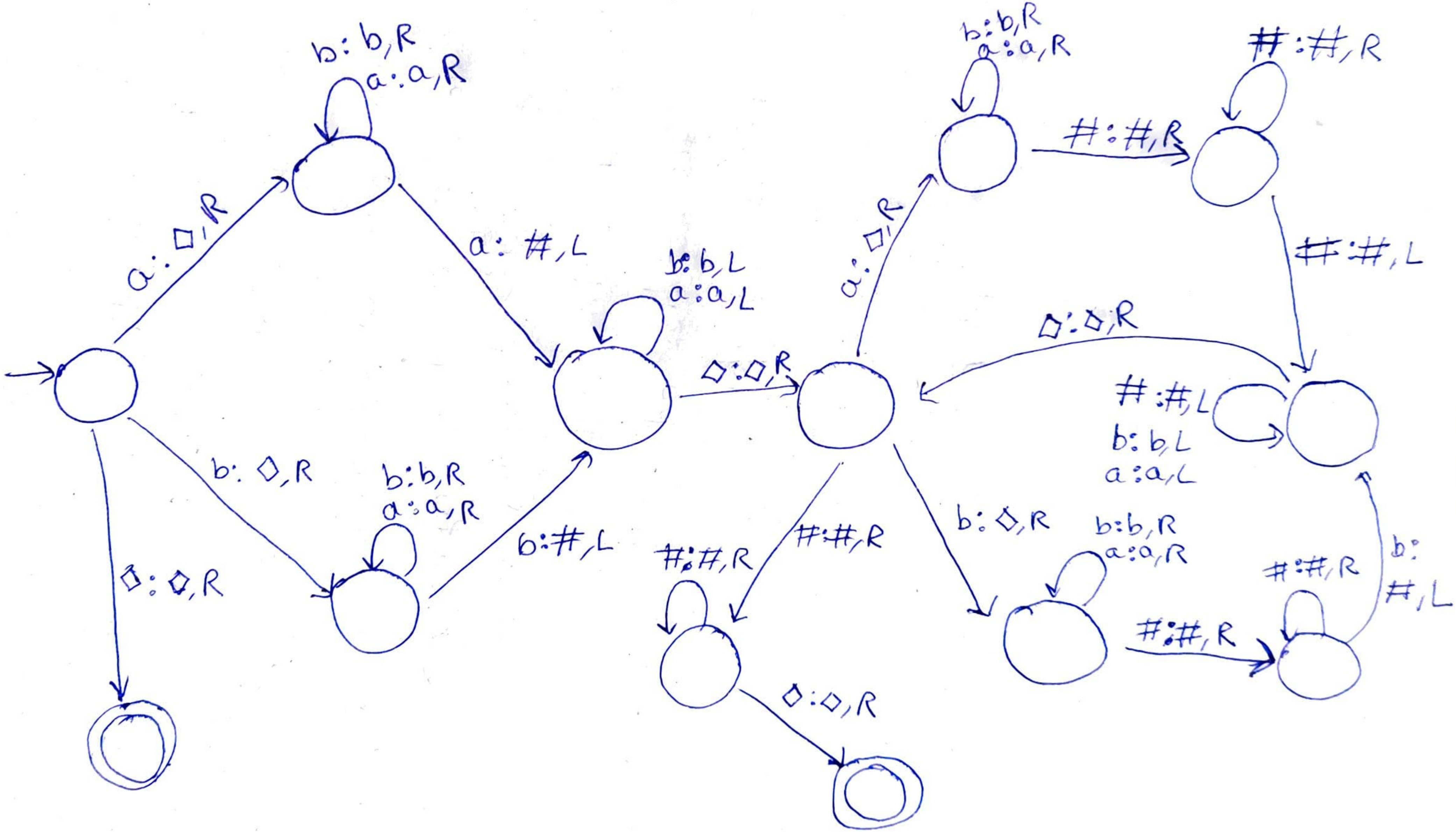
Q3)

a) Find the middle, mark it. If there's a lone character in the middle (ie) The length of the input string isn't even), then reject immediately

b) Bounce back and forth between the beginning of the first w and the beginning of the second marking off characters if they match and rejecting if they dont.

c) If we get to the end of the w's and everything has matched, accept.

P.T.O


b:b,R
a:a,R
a:□,R
a: #,L
b:b,L
a:a,L
□:□,R
b: □,R
b:b,R
a:a,R
b:#,L
□:□,R
b:b,R
a:a,R
a:□,R
#:#,R
#:#,R
#:#,L
□:□,R
#:#,L
b:b,L
a:a,L
#:#,R
#:#,R
b:□,R
b:b,R
a:a,R
#:#,R
#:#,R
b:
#,L
□:□,R