Sheet 11

Digdarshan Kunwar

December 2018

Problem 11.1

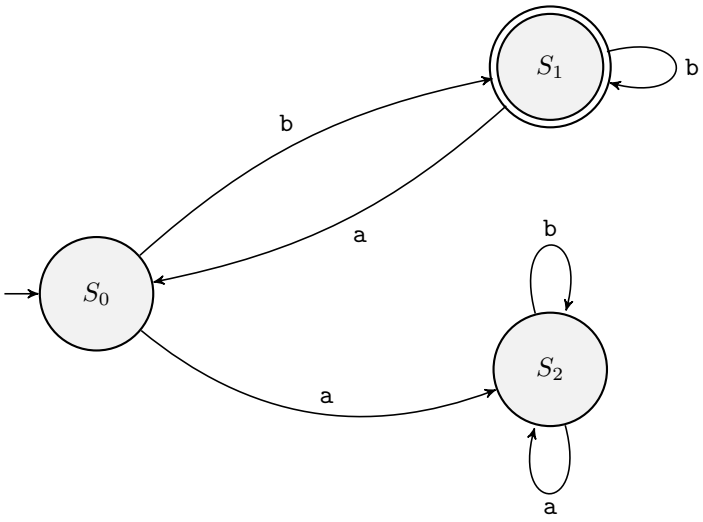
a)

The definition of finite state machine
Here,
$FSM\ \ (\sum, S, s_o, \delta, F)$

$\sum = \{a, b\}$
S=$\{S_o, S_1, S_2\}$
$\mathbf{s_0} = \{S_o\}$
$\mathbf{F} = \{S_1\}$

$$\delta = \{((S_o, b), S_1), (S_o, a), S_2),$$
$$(S_1, a), S_o), (S_1, b), S_1),$$
$$(S_2, a), S_2), (S_2, b), S_2)\}$$

b)



c)

```
{-
CH08-320101_f2018
Problem 11.1 c) hs
Digdarshan Kunwar
d.kunwar@jacobs-university.de
-}
{-
```

```
Using results from HomeWork We can construct the following program
-}

data State = S0 | S1 | S2 | S3

accepts :: State -> String -> Bool

{-these are the equavalent delta functions from the documentation of the FSM-}
accepts S0 ('b':xs) = accepts S1 xs
accepts S0 ('a':xs) = accepts S2 xs
accepts S1 ('b':xs) = accepts S1 xs
accepts S1 ('a':xs) = accepts S0 xs
accepts S2 ('a':xs) = accepts S2 xs
accepts S2 ('b':xs) = accepts S2 xs


accepts S1 [] = True
accepts _ _ = False

fsm :: String -> Bool

{-Inital State-}
fsm = accepts S0
```

d)

Here:
Grammar required to generate the given set of rules:

$(N, \sum, P, S)$

$\sum = \{a, b\}$
$N = S, T, U$
$P = \{S \to bT, T \to bT, T \to aX, X \to bT, X \to b, T \to \varepsilon\}$

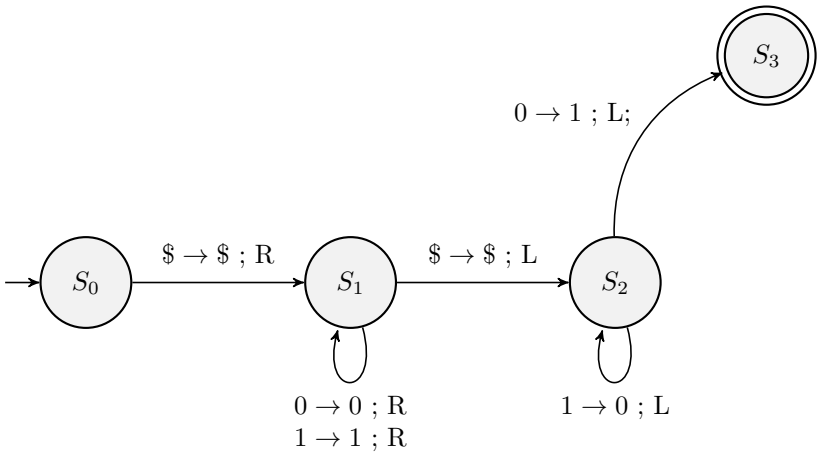
Problem 11.2

a)

Initially
where x can be 0 or 1

↓					
$	0	x	x	x	$

Here the logic is simple the machine initially starts at the first with the pointer(head) at **$** so it moves right until it reaches a the ending **$** that is at the rightmost part of the Turing tape.
So when the (head)pointer reads **$** then it moves left and checks if the digit is 1 or 0.
If the digit is 0 then it rewrites it to 1 and moves to accepting(Final) state.
If the digit is 1 then it rewrites it to 0 and moves left and to the same state.So the pointer(head) moves left changing 1 to 0 until the head reads 0.

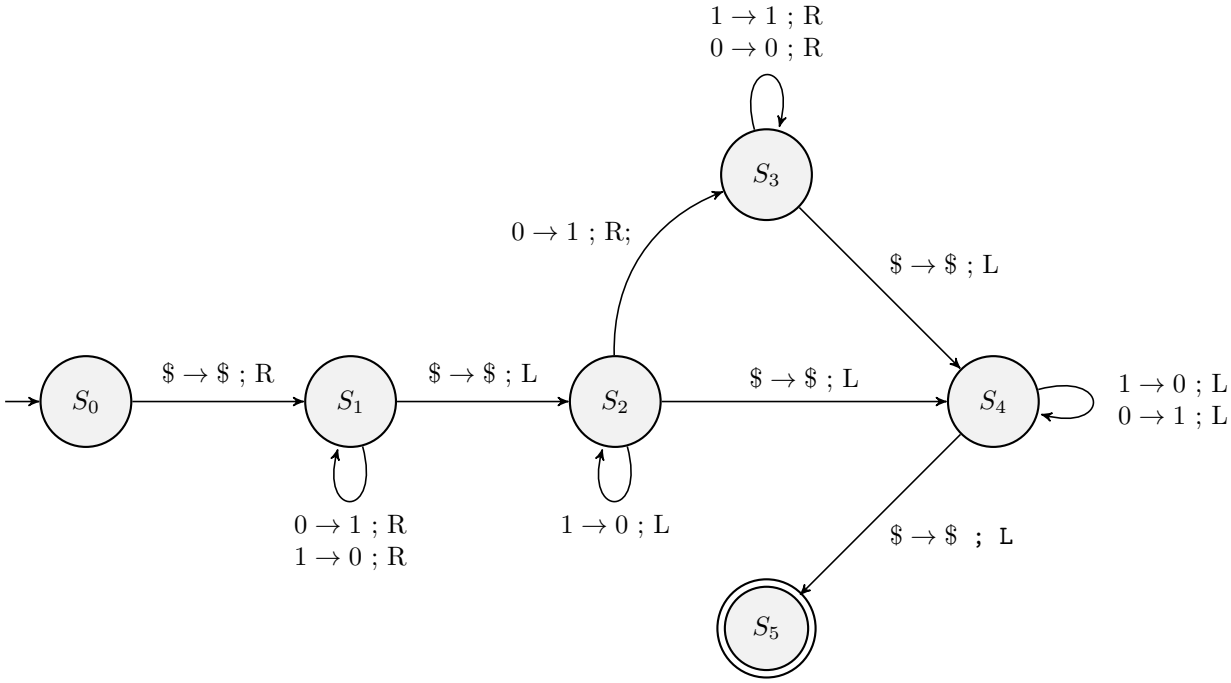


b)

Initially
where x can be 0 or 1

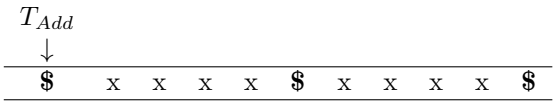
↓					
$	0	x	x	x	$

The logic is similar as that of question **a)** but in its initial movement form the leftmost **$** to the rightmost **$** the digits on the Turing tape are flipped (1 to 0 and 0 to 1).
So it adds one to the binary number and reaches state 3.
So after adding one,all the bits need to be flipped again.So the head moves to the right most **$** and start flipping the digits (1 to 0 and 0 to 1) to it's way to the leftmost **$** by moving left everytime until **$** in the leftmost side is reached.

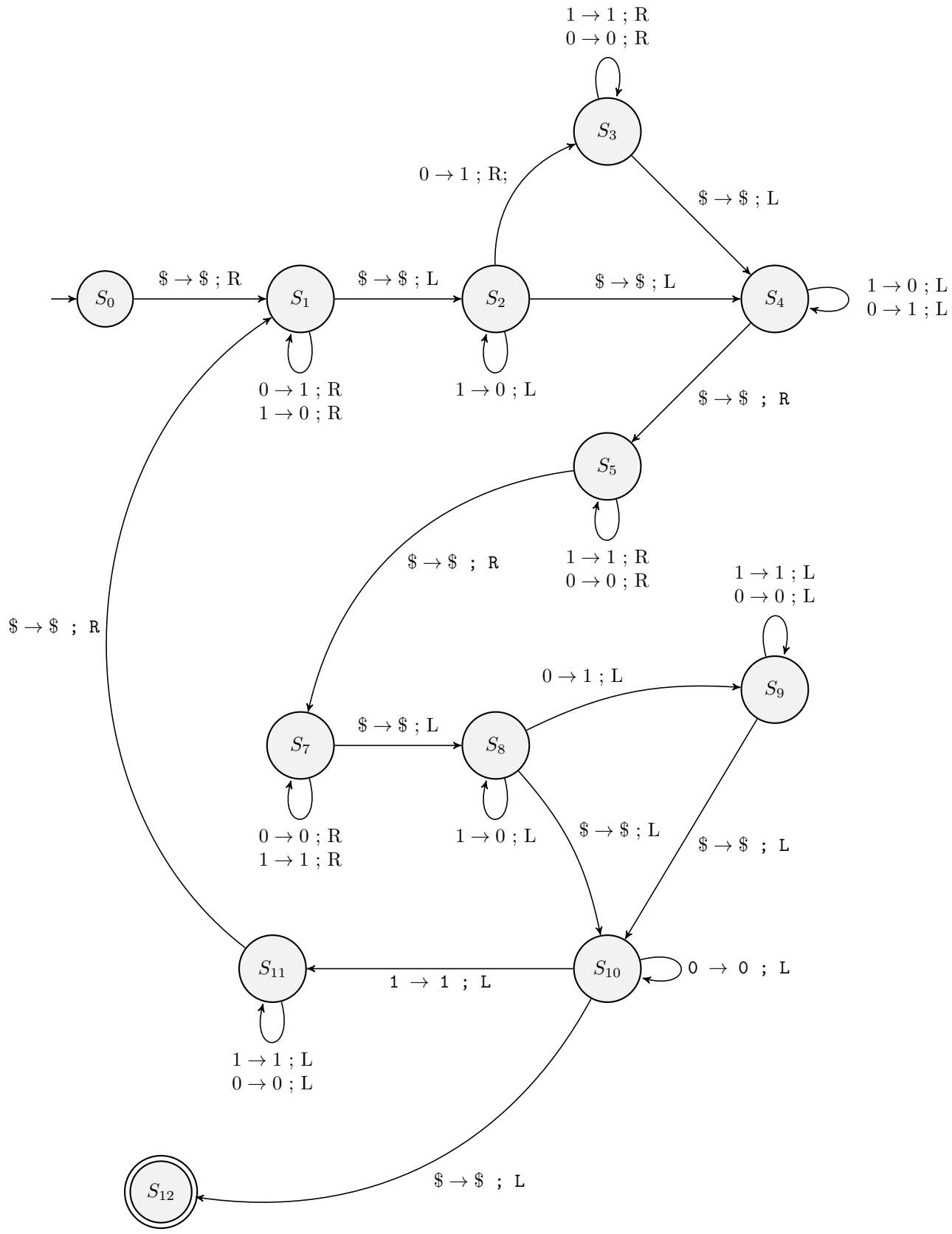


c)

Initially
where x can be 0 or 1



Here,
The logic is that once the number in the left part decreases then the process starts to increase the number by one in the second part.
So every time until the binary number between the first $ x x x x$ are not $0 0 0 0 $ the Turing machine continuously decreases 1 to the first side and increases 1 to the second side.
So the final state is reached when all the digits in the left side are $0 0 0 0 $.
Until then the loop runs substrating one from the first part and adding one in the second.
So we combine the turing machine in **a)** and **b)**.


S_0
S_1
S_2
S_3
S_4
S_5
S_7
S_8
S_9
S_{10}
S_{11}
S_{12}
$\$ \to \$$; R
$\$ \to \$$; L
$1 \to 1$; R
$0 \to 0$; R
$0 \to 1$; R;
$1 \to 0$; L
$0 \to 1$; R
$1 \to 0$; R
$1 \to 1$; L
$0 \to 0$; L
$0 \to 0$; R
$1 \to 1$; R
$0 \to 1$; L
$1 \to 0$; L
$\$ \to \$$; R
1 → 1 ; L
0 → 0 ; L