COLLEGE OF COMUTING

DEPARTMENT COMPUTE SCIENCE

Complexity Theory Group Assignments

Course Code **Cosc4132**

<u>**Members**</u> <u>**Id**</u>

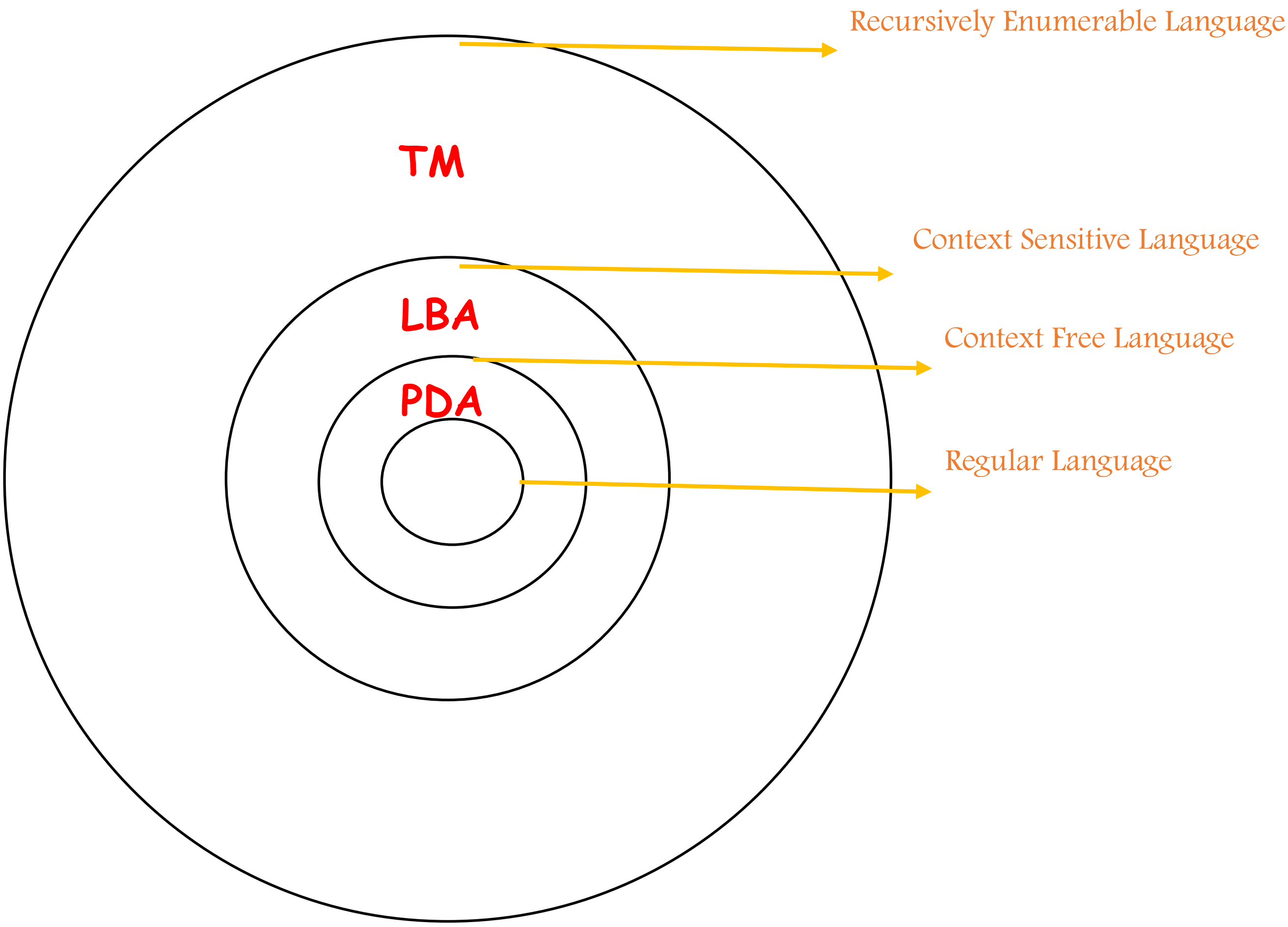
1. Surafel Takele---1319/10
2. Yalewgiz Tadegegn---1345/10
3. Tesfa Tadesse--1329/10
4. Sisay Golye--1313/10
5. Samule Degu--1296/10
6. Habtewold Tagafew---1281/09

Submitted To : msc Yeharer work

Submitted Date 11/06/2013

1. **Mention a TM example for acceptance and rejection cases other than the examples given in the slide.**

Introduction-TM



Opration on the tape

- Read/Scan symbol at the tape head
- Update/Write a symbol at the tape head
- Move the tape head one step left
- Move the tape head one step right

Rules of Opration-1

At each step of the computation:

- Read/Scan the current symbol
- Update/Write the same cell
- Move exactly one cell either left or right

Rules of Opration-2

- Control is with a sort of FSM
- Initial State
- Final States:(There are two final states)
    1) THE ACCEPT STATE
    2) THE REJECT STATE
- Comutation can be either
    1) HALT and ACCEPT
    2) HALT and REJECT
    3) LOOP(the machine fails to HALT)

Design a TM that accepts

{0"1 " n>=2}.

Solution

We require the following moves:

(a) If the leftmost symbol in the given input string w is 0, replace it by x and move right till we encounter a leftmost 1 in w. Change it to y and move backwards.

(b) Repeat (a) with the leftmost 0. If we move back and forth and no 0 or 1 remains. move to a final state.

(c) For strings not in the form 0"1", the resulting state has to be no final.

Keeping these ideas in our mind, we construct a TM M as follows:

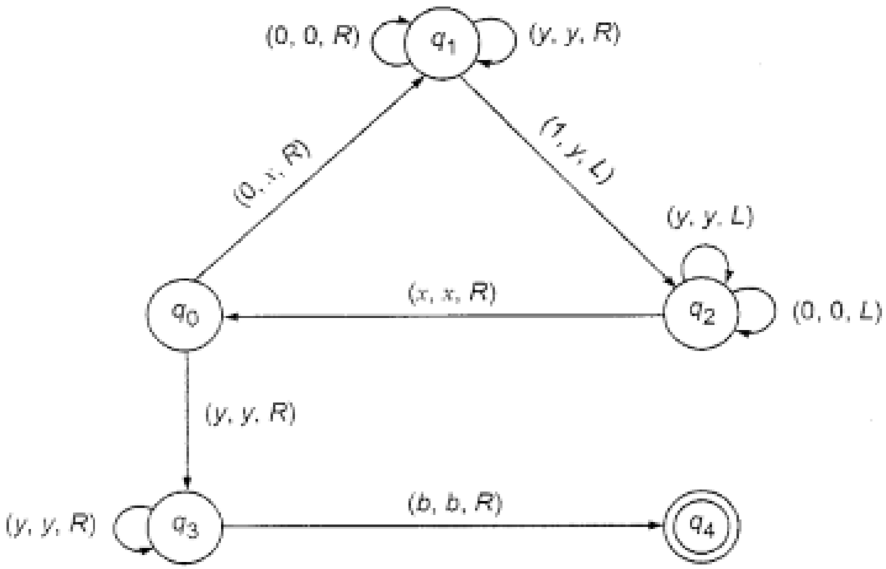
M =(Q, ∑, , Γ,δ, q_0, b, F)

Q = {q_0,q_1,q_2,q_3,q_f)

F ={ q_f }

∑, ={0,1}

Γ = {O, 1, x, y, b}



$$q_0 0011 \vdash xq_1 011 \vdash x0q_1 11 \vdash xq_2 0y1$$
$$\vdash q_2 x0y1 \vdash xq_0 0y1 \vdash xxq_1 y1 \vdash xxyq_1 1$$
$$\vdash xxq_2 yy \vdash xq_2 xyy \vdash xxq_0 yy \vdash xxyq_3 y$$
$$\vdash xxyyq_3 = xxyyq_3 b \vdash xxyybq_4 b$$

Hence 0011 is accepted by M.

$$q_0 010 \vdash xq_1 10 \vdash q_2 xy0 \vdash xq_0 y0 \vdash xyq_3 0$$

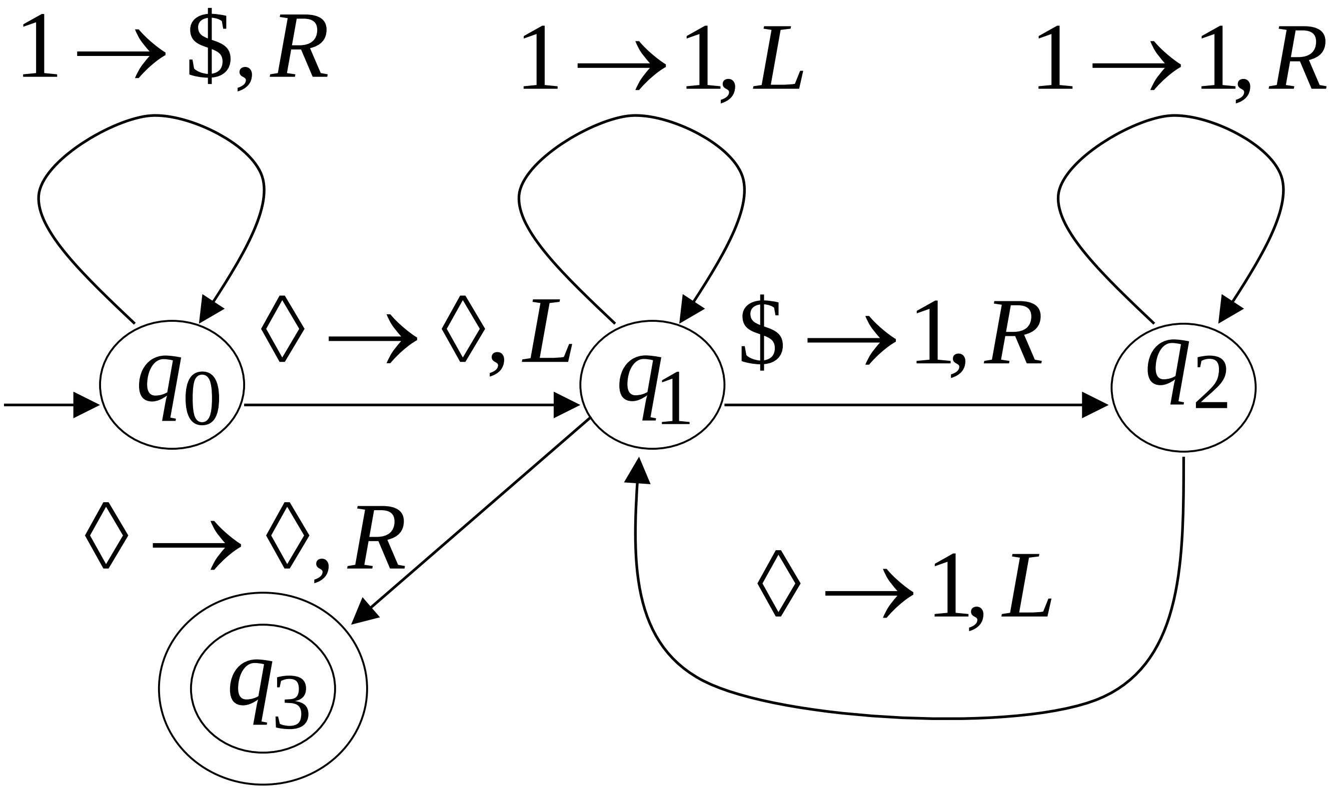
As $\delta(q_3, 0)$ is not defined, M halts. So 010 is not accepted by M.

2. Is the function f(x)=2X is computable?

A function *f* : x → x is total (or just function) when *f*(x) is defined for every x

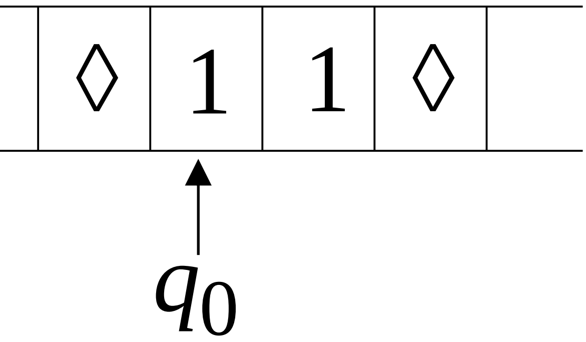
Turing Machine for

$$f(x) = 2x$$

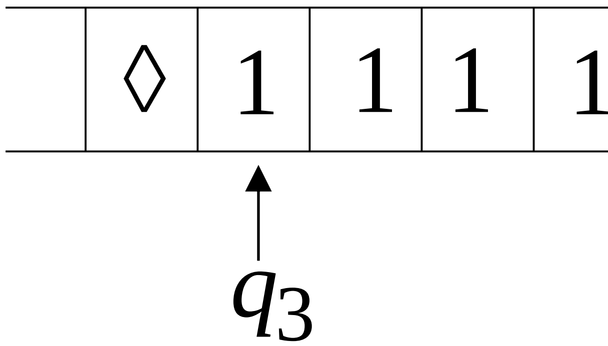


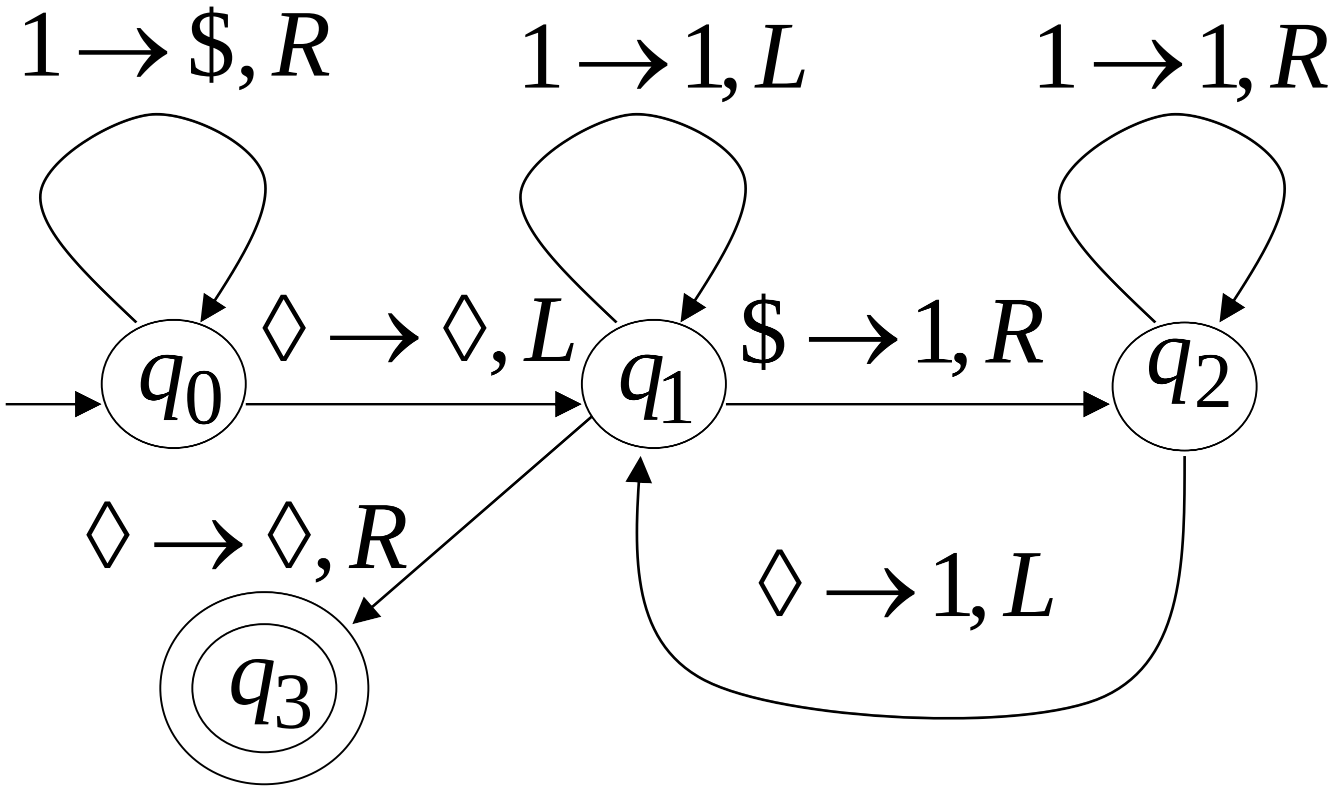
Example

Start



Finish





3. **Construct a TM to:**
   a) subtract two Unary Numbers

Problem-1: Draw a Turing machine which subtract two numbers m and n,where m is less then n.

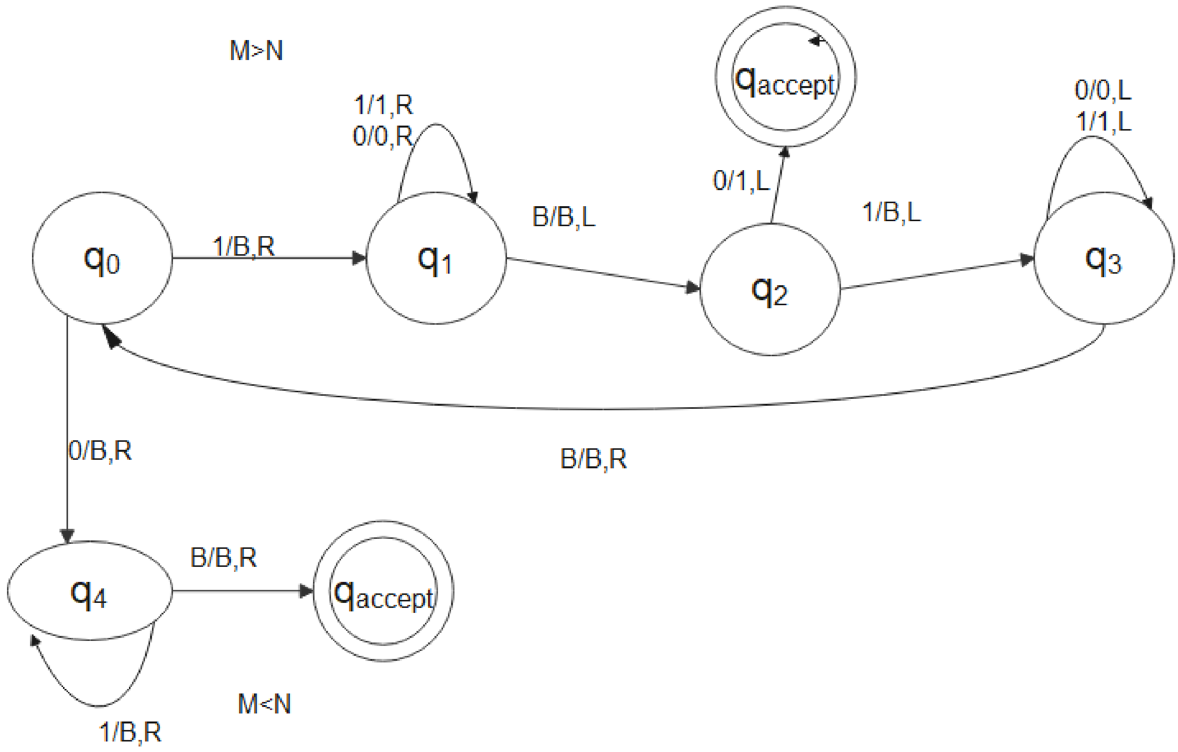
Steps:

- **Step-1.** If 1 found convert 1 into X and go right then convert all 1's into 1's and go right.
- **Step-2.** Then convert 0 into 0 and go right then convert all B into B and go right.
- **Step-3.** Then convert 1 into B and go left then convert all B into B and go left.
- **Step-4.** Then convert 0 into 0 and go left then convert all 1's into 1's and go left then convert all B into B and go right and repeat the whole process.
- **Step-5.** Otherwise if 0 found convert 0 into 0 and go right then convert all B into B and go right then convert 0 into 0 and go left and then **stop the machine.**

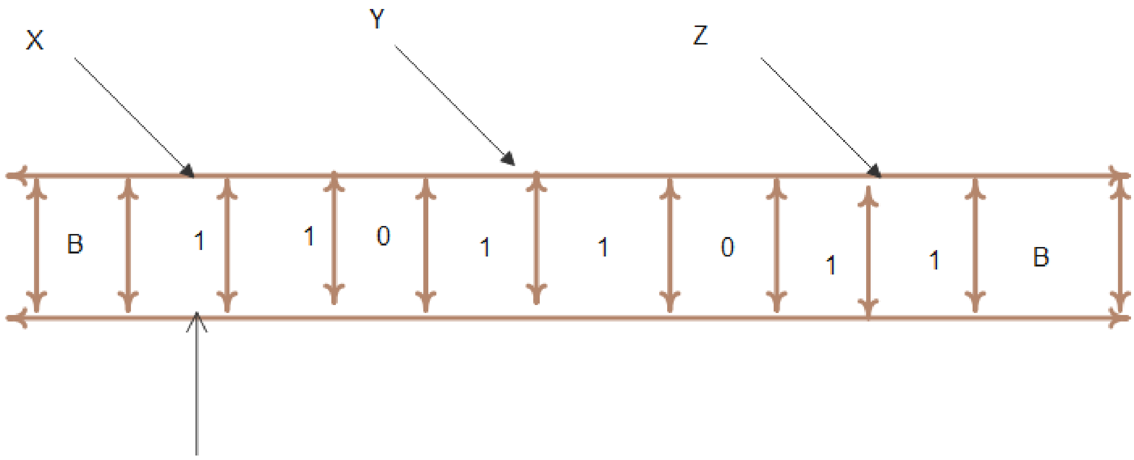
Problem-2: Draw a Turing machine which subtract two numbers m and n, where m is greater then n.

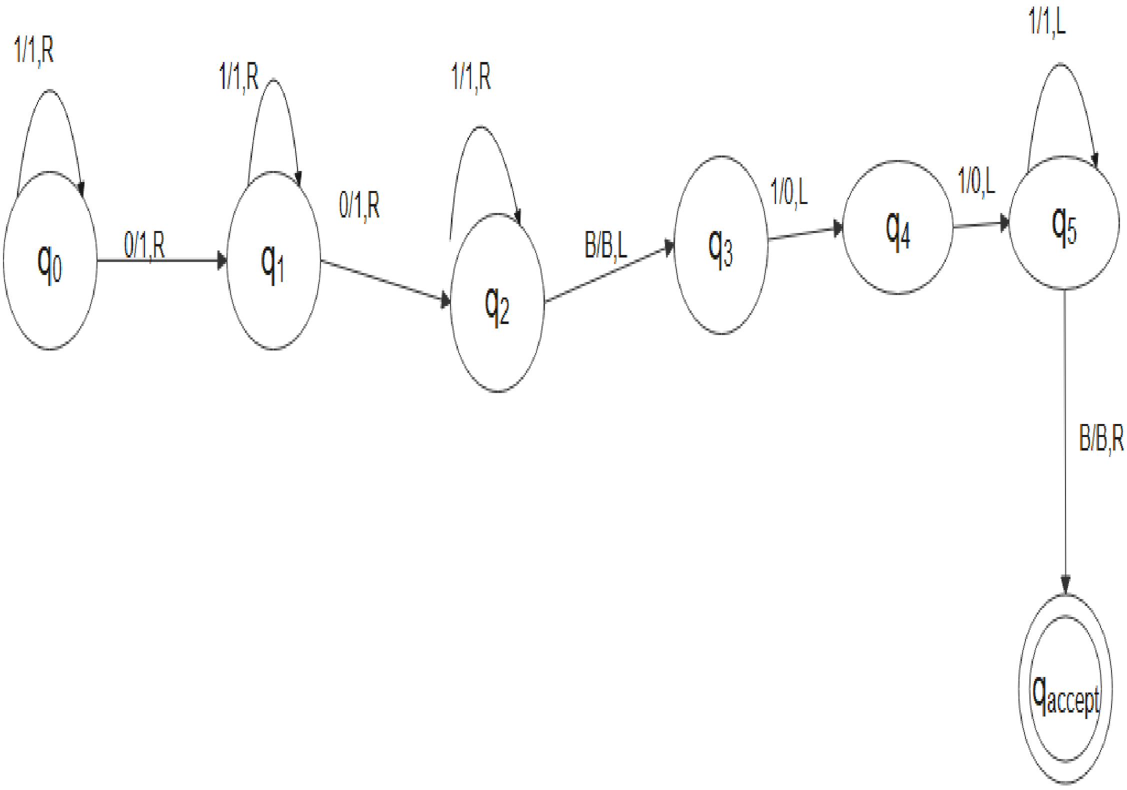
Steps:

- **Step-1.** If 1 found convert all 1's into 1's and go right then convert 0 into 0 and go right
- **Step-2.** If B found then convert all B into B and go right or if 1 found then convert 1 into B and go left and go to next step otherwise go to 5th step
- **Step-3.** Then convert all B into B and go left then convert 0 into 0 and go left
- **Step-4.** Then convert all 1's into 1's and go left then convert X into X and go right then convert 0 into B and go right and repeat the whole process
- **Step-5.** Otherwise if B found convert B into B and go left then convert all B into B and go left then convert 0 into B and go left and then **stop the machine.**



b) evaluate the function f(x)=X+Y+Z, where X,Y and Z are all unary numbers.





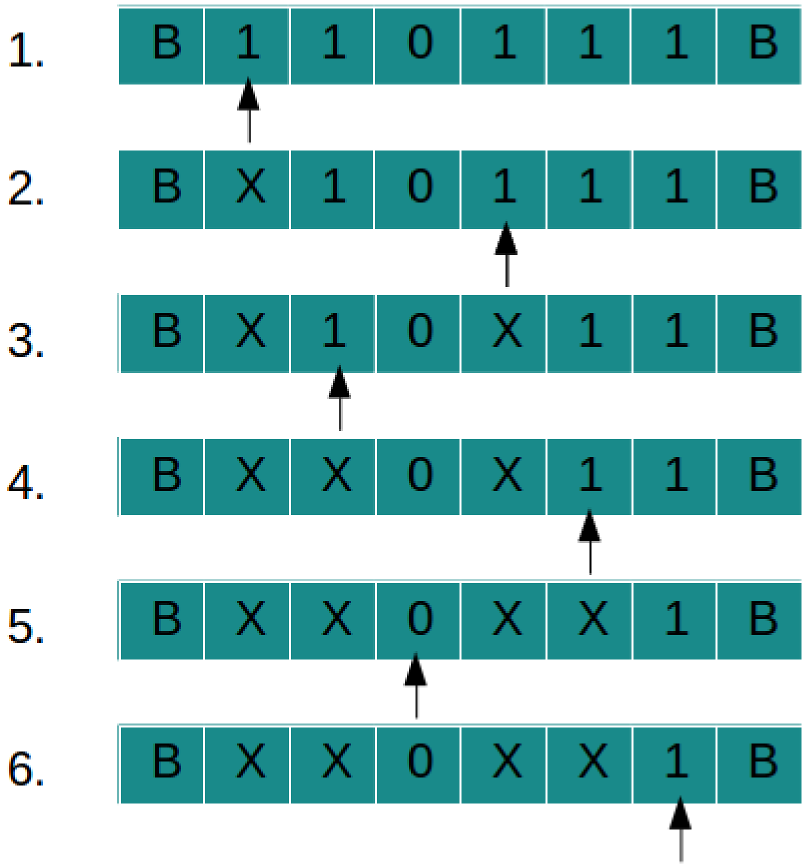
c) compare two string.

Turing machine as Comparator

Approach for Comparator

1. Matching two numbers by comparing '1's
2. Example: 11101111 (3 and 4): compare '1's by marking them 'X' and 'Y' respectively
3. If '1's are remaining in left of '0' and in right of '0', '1' are finished, then formar is greater
4. If '1's are remaining in right of '0' and in left of '0', '1' are finished, then later is greater
5. If both '1' are finished then numbers are equal

TAPE movement for string "110111":



Explanation of TAPE movement

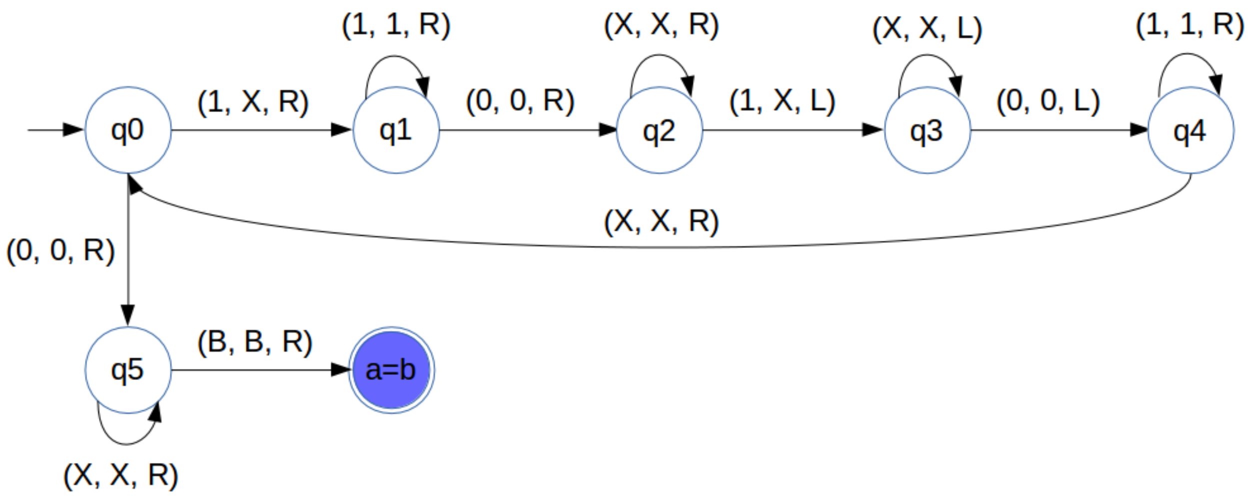
1. Input is given as "110111" (2 and 2)
2. Scan string from left to right
3. Mark '1' as 'X' and then move to right
4. Reach right of '0' and mark '1' as 'X' and move left
5. Reach 'X' in left of '0' and move one step right
6. Again mark '1' as 'X' and then move to right
7. Reach right of '0' and pass 'X', mark '1' as 'X' and move left
8. Reach 'X' in left of '0'(passing 'Y', '0' and '1')and move one step right
9. As '0' is there after 'X' that means all '1's are finished before '0'
10. Check for '1' in right of '0'
11. Pass '0', 'X' and there is '1' remaining that means second number is greater than first one
12. And final state for this will be "a<b"

State Transition Diagram

We have designed state transition diagram for adder as follows:

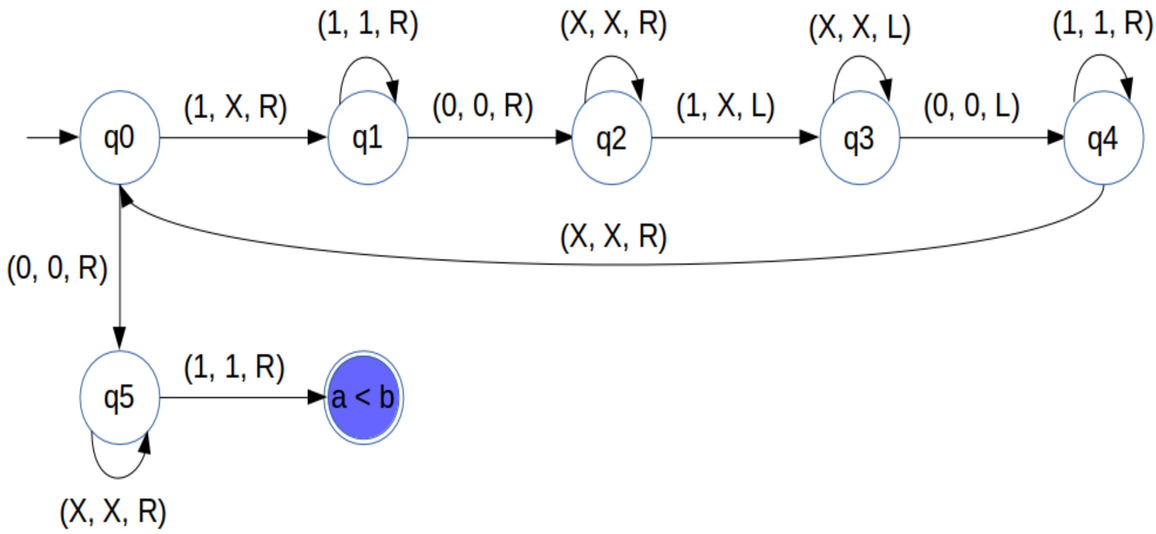
Comparator for a = b

1. This concept is similar to $a^n b^n$



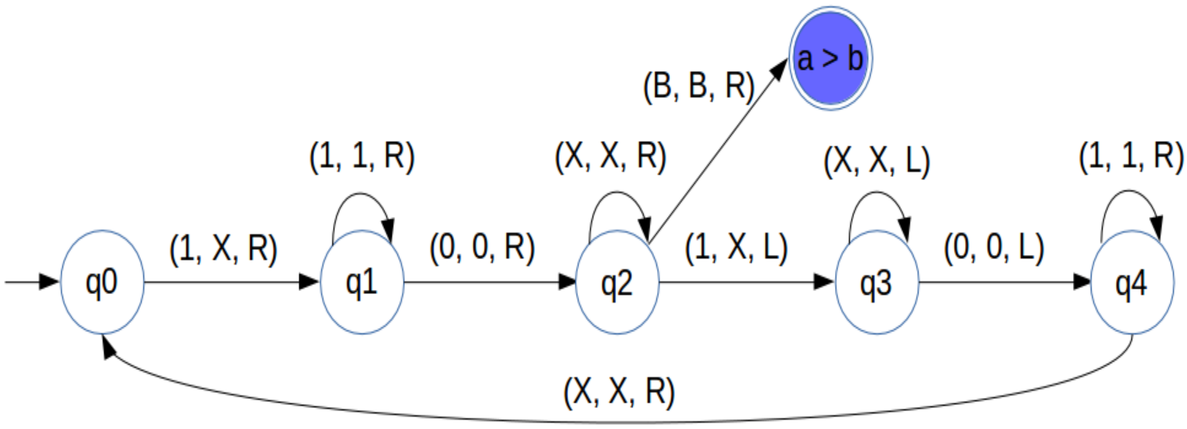
Comparator for a < b

2. When all '1's are finished in left of '0'

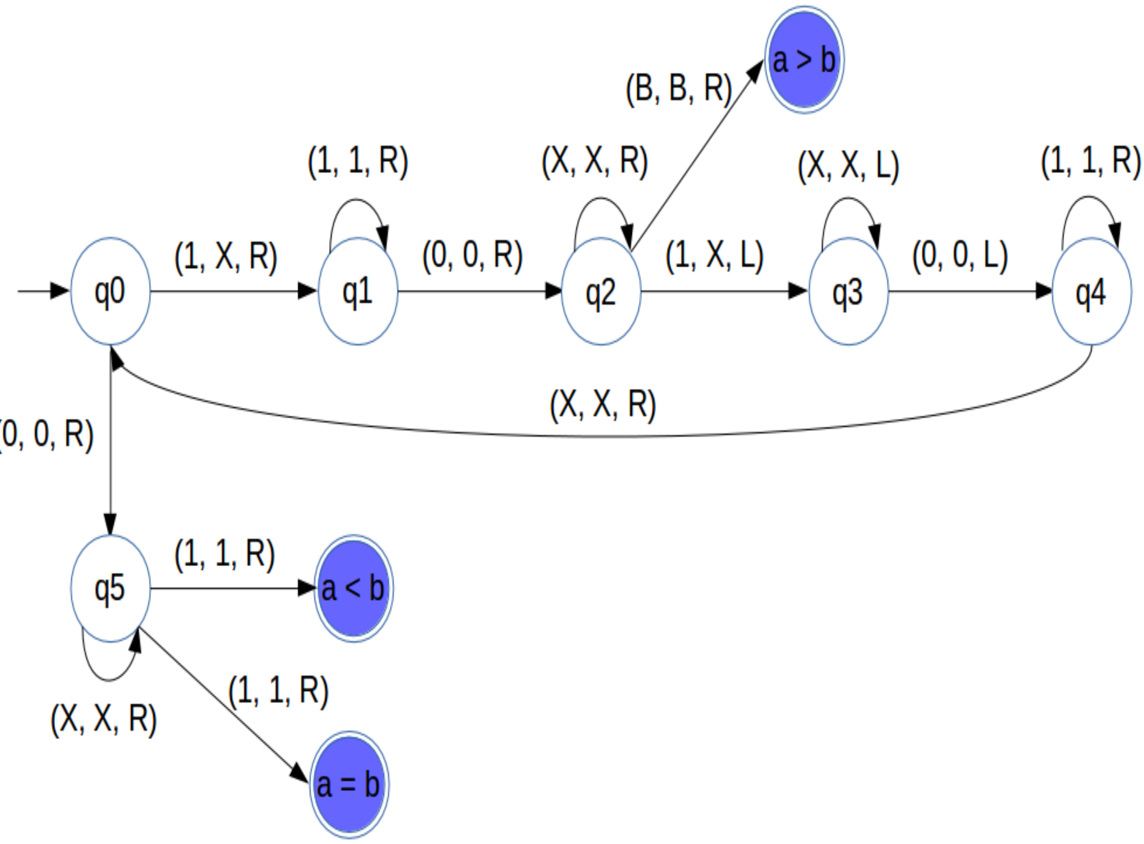


Comparator for a > b

3. When all '1's are finished in right of '0'

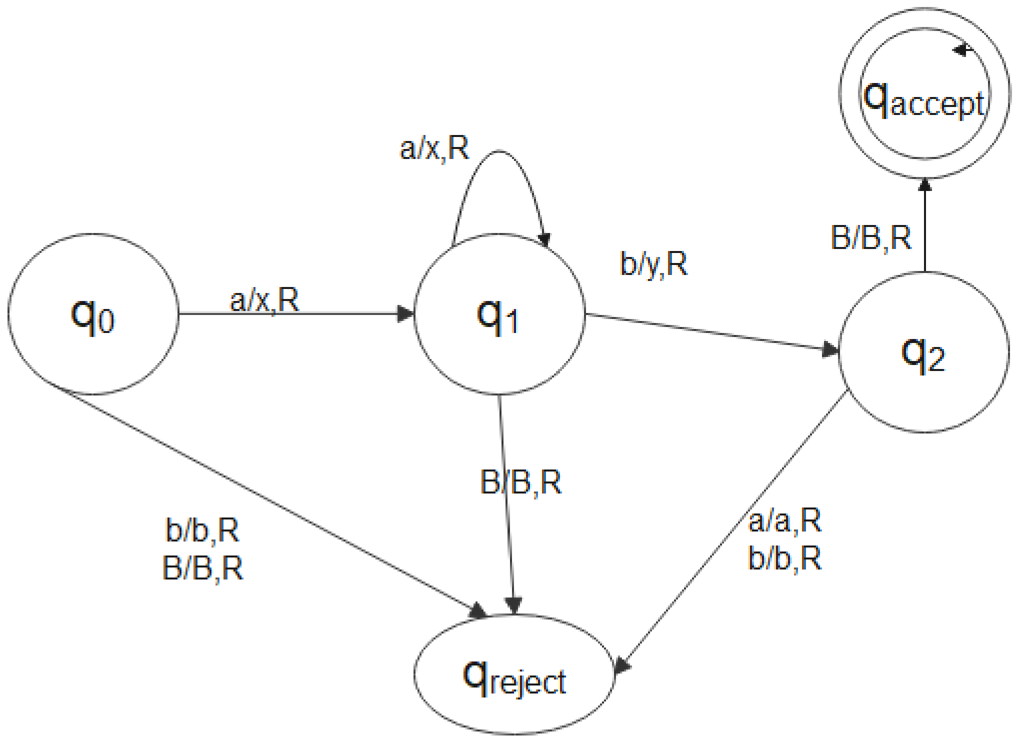


4. **Final version of Comparator**



4. Construct a TM that accepts the language

a) {ab, aab, aaab, aaaab, ….}



b) (a^n, b^n, c^n)

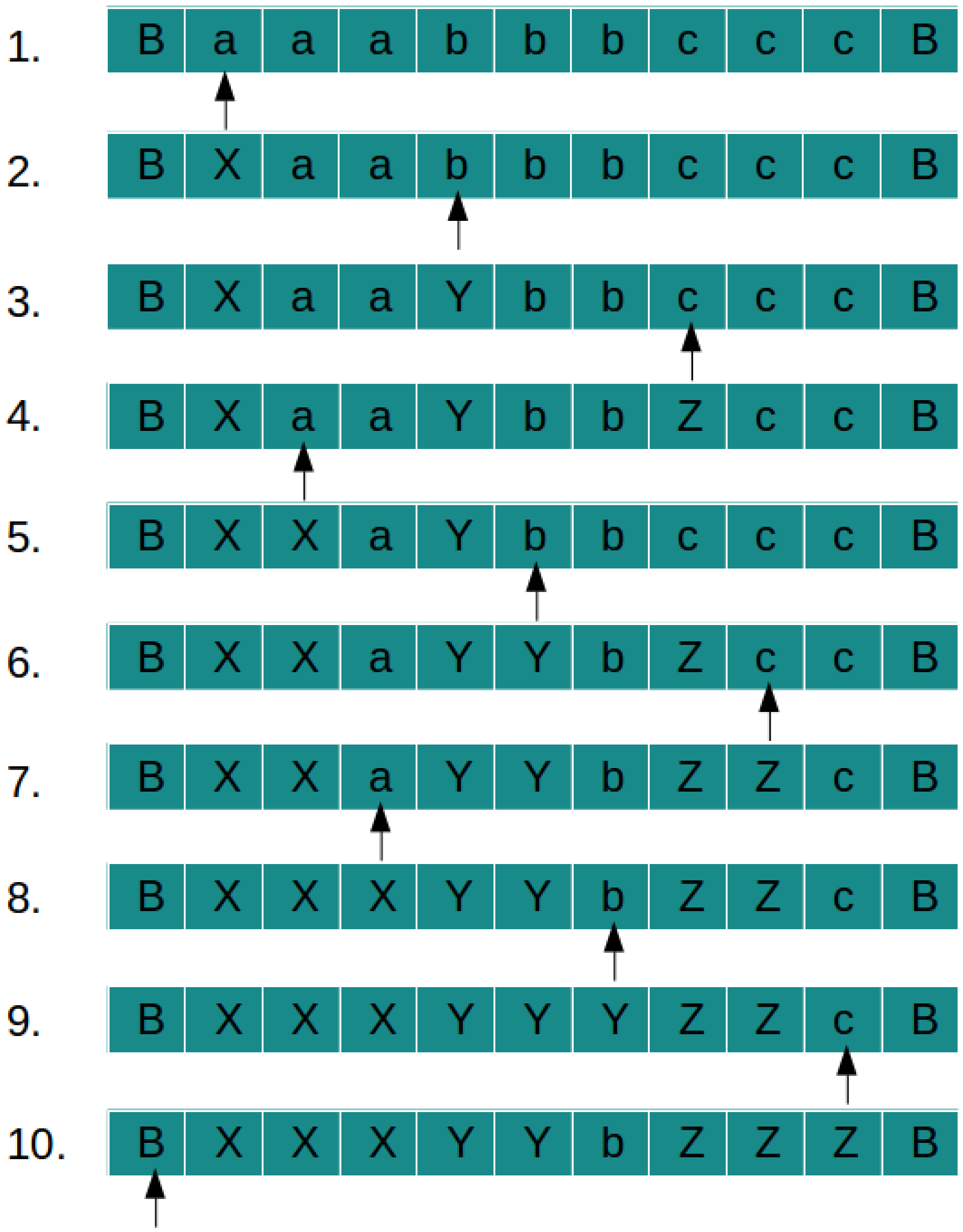
Turing machine for $a^n b^n c^n \mid n \geq 1$

Previously we have seen example of turing machine for $a^n b^n \mid n \geq 1$
We will use the same concept for $a^n b^n c^n \mid n \geq 1$ also.

Approach for $a^n b^n c^n \mid n \geq 1$

1. Mark 'a' then move right.
2. Mark 'b' then move right
3. Mark 'c' then move left
4. Come to far left till we get 'X'
5. Repeat above steps till all 'a', 'b' and 'c' are marked
6. At last if everything is marked that means string is accepted.

TAPE movement for string "aaabbbccc":



Explanation of TAPE movement

Step 1-2

1. Input is given as "aaabbbccc" (scan string from left to right)

2. Mark 'a' as 'X' and move one step right
3. Reach unmarked 'b' and pass every 'a' and 'Y'(in case) on the way to 'b'

Step 3-4

4. Mark 'b' as 'Y' and move one step right
5. Reach unmarked 'c' and pass every 'b' and 'Z'(in case) on the way to 'c'
6. Mark 'c' as 'Z' and move one step left cause now we have to repeat process
7. Reach unmarked 'X' and pass every 'Z', 'b', 'Y', 'a' on the way to 'X'
8. Move to 'a' and repeat the process

Step 5-9

9. Step 1-4 are repeated

Step 10

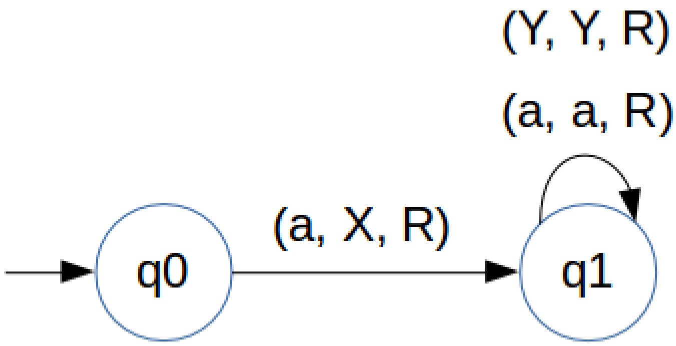
10. When TAPE header reaches 'X' and next symbol is 'Y' that means 'a's are finished
11. Check for 'b's and 'c's by going till BLANK(in right) and passing 'Y' and 'Z' on the way
12. If no 'b' and 'c' are not found that means string is accepted

State Transition Diagram

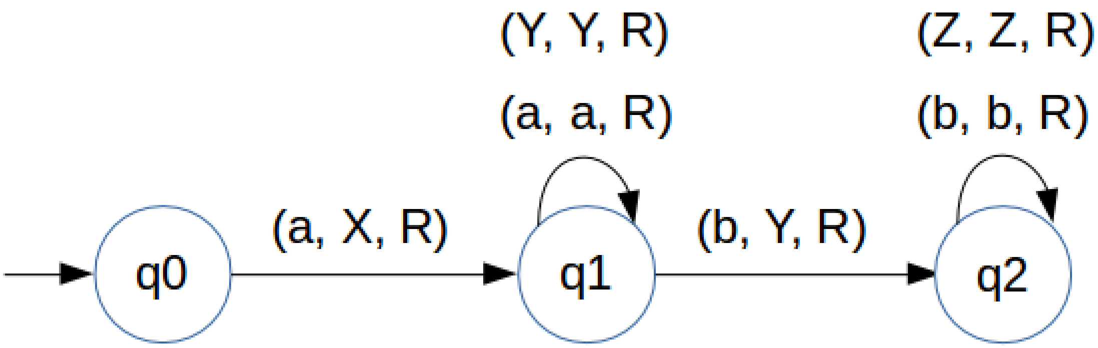
We have designed state transition diagram for $a^n b^n c^n \mid n \geq 1$ as follows:

1. Following Steps:
   a. Mark 'a' with 'X' and move towards unmarked 'b'
   b. Move towards unmarked 'b' by passing all 'a's

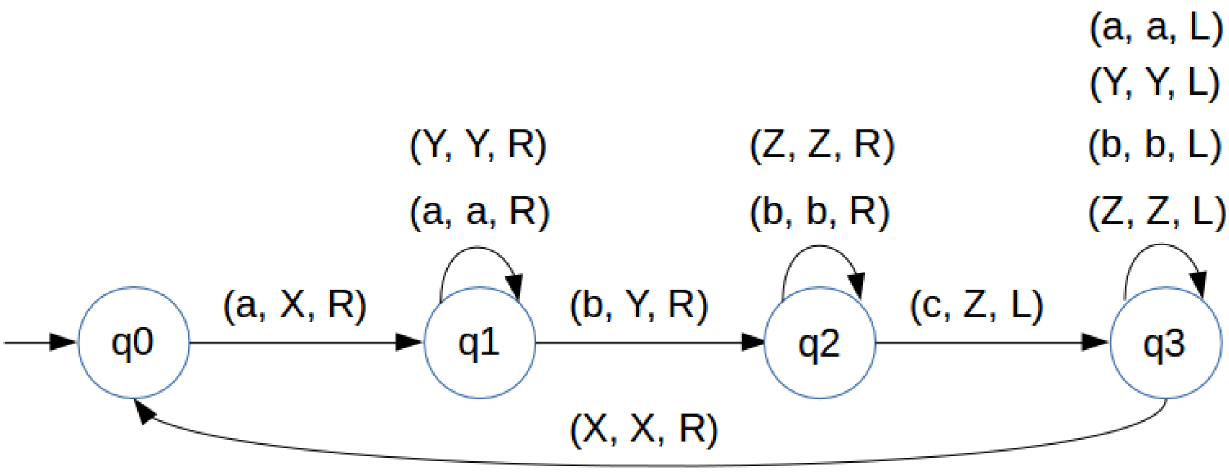
c. To move towards unmarked 'b' also pass all 'Y's if exist



2. Following Steps:
   a. Mark 'b' with 'Y' and move towards unmarked 'c'
   b. Move towards unmarked 'c' by passing all 'b's
   c. To move towards unmarked 'c' also pass all 'Z's if exist

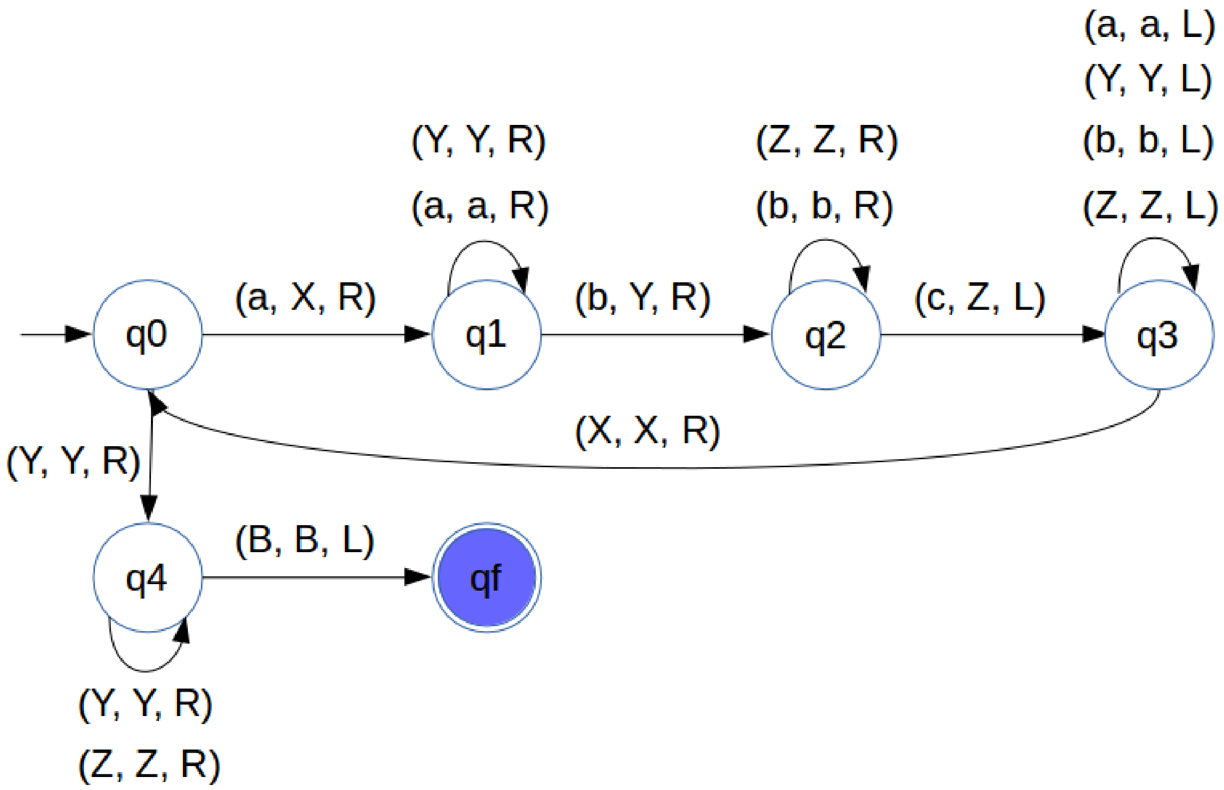


3. Following Steps:
   a. Mark 'c' with 'Z' and move towards first 'X' (in left)
   b. Move towards first 'X' by passing all 'Z's, 'b's, 'Y's and 'a's
   c. When 'X' is reached just move one step right by doing nothing.



4. To check all the 'a's, 'b's and 'c's are over add loops for checking 'Y' and 'Z' after "we get 'X' followed by 'Y'"
   To reach final state(qf) just replace BLANK with BLANK and move either

direction


(a, a, L)
(Y, Y, L)
(Y, Y, R)
(Z, Z, R)
(b, b, L)
(a, a, R)
(b, b, R)
(Z, Z, L)
(a, X, R)
(b, Y, R)
(c, Z, L)
q0
q1
q2
q3
(X, X, R)
(Y, Y, R)
(B, B, L)
q4
qf
(Y, Y, R)
(Z, Z, R)