A PROJECT ON

TURING MACHINE SIMULATOR

A DISSERTATION
SUBMITTED FOR THE MID TERM PROJECT EVALUATION (MTE) FOR THE COMPLETION OF COURSE THEORY OF COMPUTATION

MADE BY:
VISHRUTH KHARE (2K18/CO/393)
VISHWAS AGARWAL (2K18/CO/395)

UNDER THE SUPERVISION OF
Mr. RAHUL KUMAR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Session: November, 2020

INTRODUCTION

A Turing Machine is a mathematical model which consists of an **infinite length tape** divided into cells on which input is given. It consists of a head which reads the input tape. A state register stores the **state** of the Turing machine. Originally it was used to monitor how can we solve a problem without using any set of rules but by using a generalized algorithm. In order to make that notion precise, we would have to set out what do we mean by Algorithm. Working on a Problem in **Formal Logic**, Alan Turing came up with a notion of any possible machine that captures how algorithm and programs work. The absolute initial Turing Machine had the following norms:

- To think of a Tape that can **accommodate all the encoding** pertaining to that particular program.
- The tape had **squares** that entail the **alphabets** taken into consideration (eg 1, 0, /)
- The machine looks at the tape **one square at a time** and comprehending the coded information based on the **set of rules** that have been defined for that formal logic.
- Simple instructions lead the machine to different states and once the **end state** has been achieved, the **final answer** is the desired result.

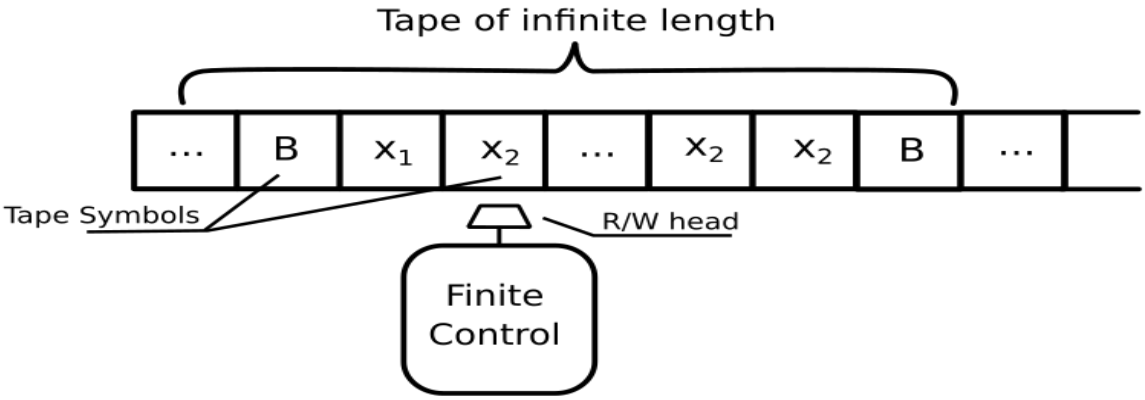
(Turing Machine Tape)

During the computation, since the final answer is not achieved we call that state the **Halting State**. Once the Halting state is finished, what's left on the tape is answer to our problem coded as 1s and 0s. Whatever a computer can do, it can be inferred by looking up the encoding on the tape. These set of functionalities acted as a blue print for the modern day computers. When a computer can do what a Turing Machine can do, we call it **Turing Complete** and that has the highest programming strength.

WORKING PRINCIPLE

The Turing Machine have three elements

1. Input/Output Tape
2. Read-Write head that is bi-directional.
3. Finite State Control



The Turing machine can be thought of as a finite automata connected to read/write head which is bidirectional i.e. can move left to right and right to left.

It has one input/output tape which is divided in many cells. At each cell only one input symbol is placed. The input and output on tape affected by the read/write head which can examine one cell in one move.

In one move, the machine examine the present symbol under the read/ write head on the tape and the present state of an automaton to determine:

- What to do with tape present symbol i.e. a new symbol to be written or no change of symbol on the tape in the cell.
- In which direction head move i.e. either the head moves one cell left (L), or one cell right (R), or stay at the same cell (N).
- The final state.

This is how the Turing Machine is able to compute large problems using a predefined set of Algorithms.

FORMAL DEFINITION

A Turing Machine is expressed as a 7-tuple (Q, T, B, $\sum$, δ, q0, F) where:

- Q is a finite set of states
- T is the tape alphabet (symbols which can be written on Tape)
- B is blank symbol (every cell is filled with B except input alphabet initially)
- $\sum$ is the input alphabet (symbols which are part of input alphabet)
- δ is a transition function which maps $Q \times T \longrightarrow Q \times T \times \{L,R\}$. Depending on its present state and present tape alphabet (pointed by head pointer), it will move to new state, change the tape symbol (may or may not) and move head pointer to either left or right.
- q0 is the initial state
- F is the set of final states. If any state of F is reached, input string is accepted.

TRANSITION FUNCTION

$$\delta(q,X) = (p,Y,R/L)$$

- q is the current state
- X is the current tape symbol pointed by the tape
- State changes from q to p

TURING MACHINE AS LANGUAGE ACCEPTOR

A Turing machine halts when it no longer has available moves.
If it halts in a final state, it accepts its input, otherwise it rejects its input.
For language accepted by M ,we define

$L(M)=\{w \varepsilon \sum+ : q0w \vdash x1qfx2$ for some qf ε F , x1 ,x2 $\varepsilon \Gamma\}$

TURING MACHINE AS TRANSDUCER

To use a Turing machine as a transducer, treat the entire nonblank portion of the initial tape as input.
Treat the entire nonblank portion of the tape when the machine halts as output. A Turing machine defines a function y = f (x) for strings x, y ε $\sum*$ if $q0x \vdash* qf\ y$
A function index is "Turing computable" if there exists a Turing machine that can perform the above task.

MULTIPLICATION

Using Turing Machine

As we know Turing Machine can be used to perform all binary, secondary and ternary operation, in this project we will be implying multiplication of two numbers and displaying the entire simulation on an Web interface.

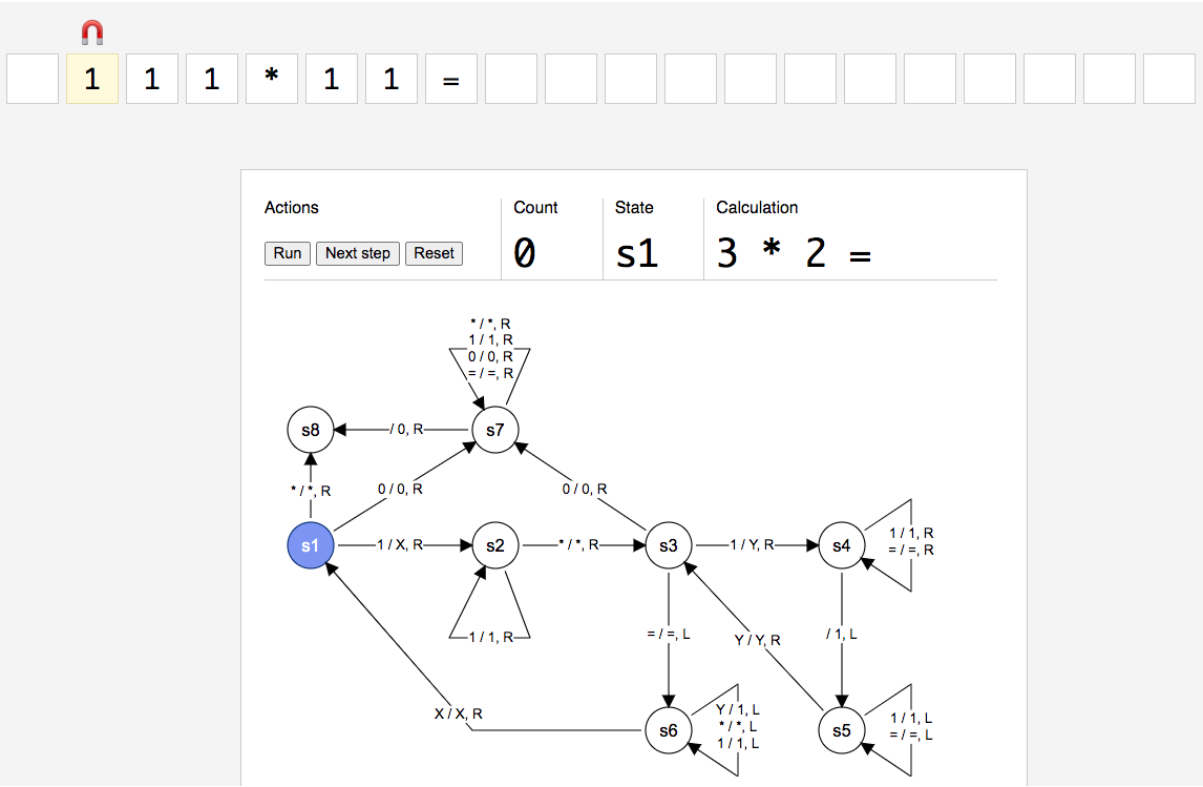
The idea of multiplication is that to perform

$$a * b = ?$$

we can perform

$$b + b + b + b \ldots\ldots \text{ a times}$$

We are simply going to create a simulation that will account for all the alphabets of the 1st number and corresponding to each digit, add a digit in the final answer until all the digits have been marked. Below is a sample of states, rules mentioned on them and the input.



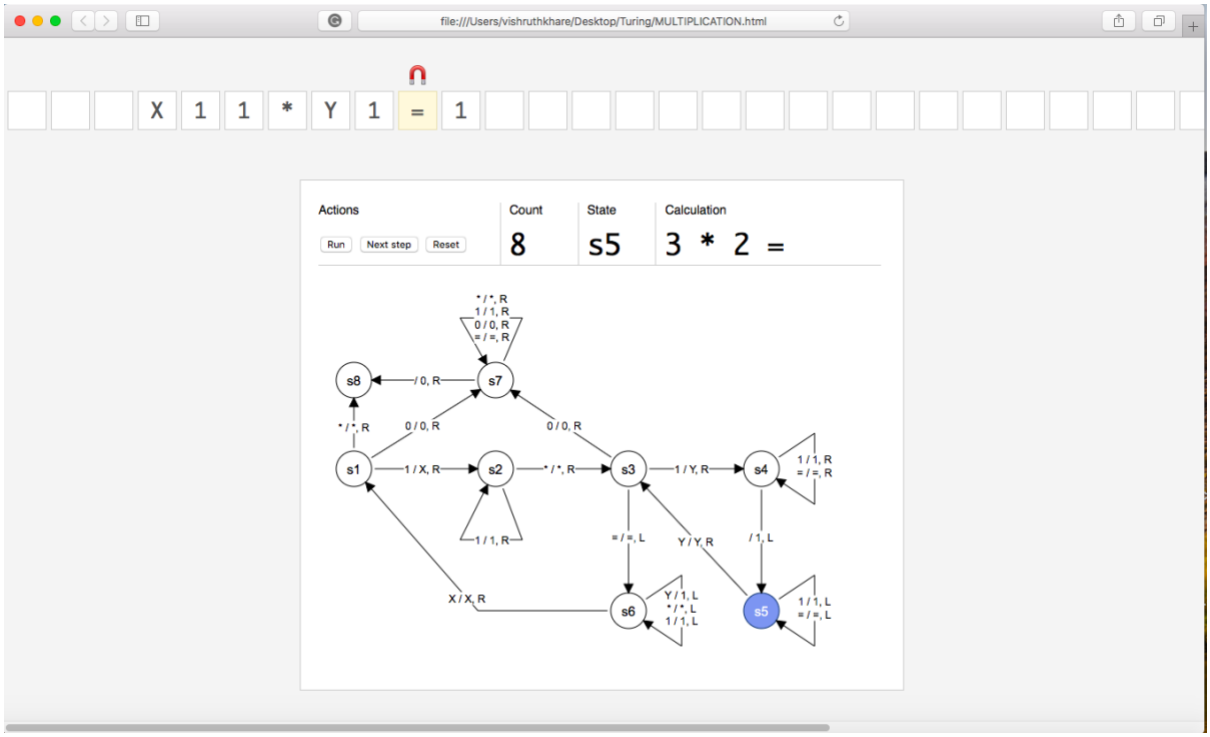
(Initial state of the machine just before the computation has begun)

Formal Logic for Multiplication

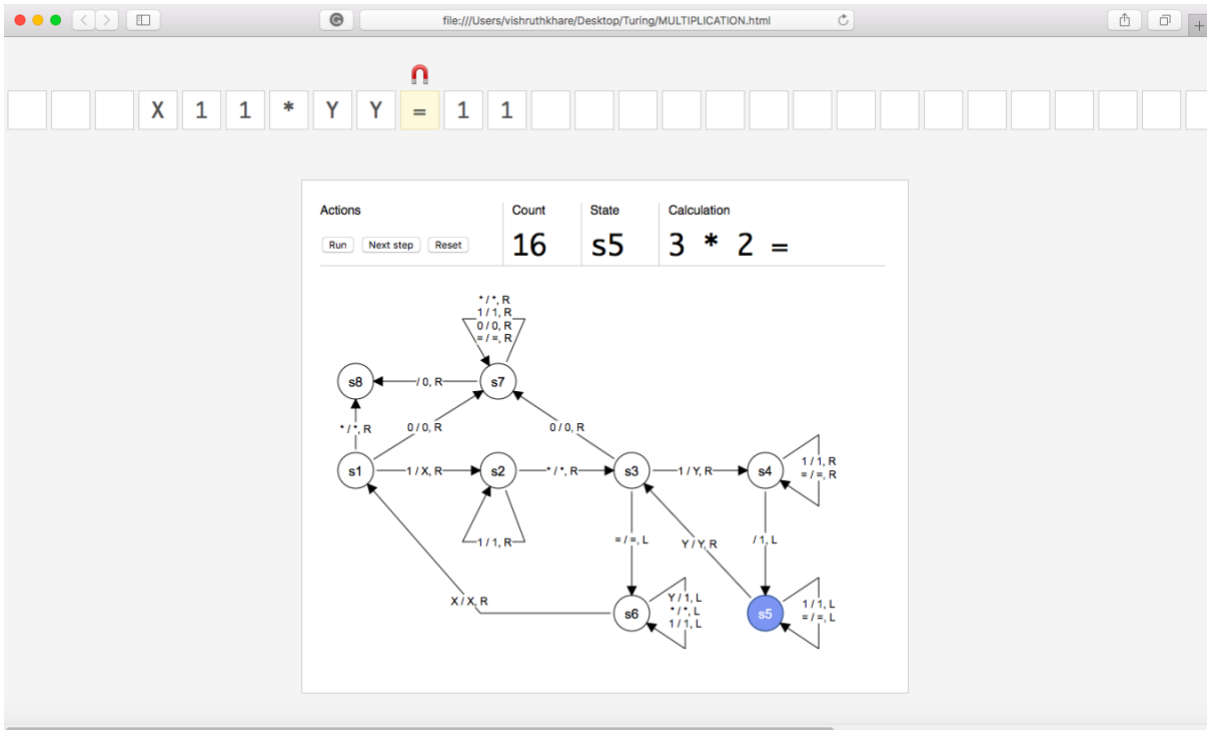
symbol	Interference
Q	S1,s2,s3,s4,s5,s6,s7,s8
T	1,*,=,X,Y,” “,0
B	“ “
∑	1,*,=
F	S8
q0	S1

We have created a set of rules as to how the Turing Machine will respond to each state. The algorithm is as follows:

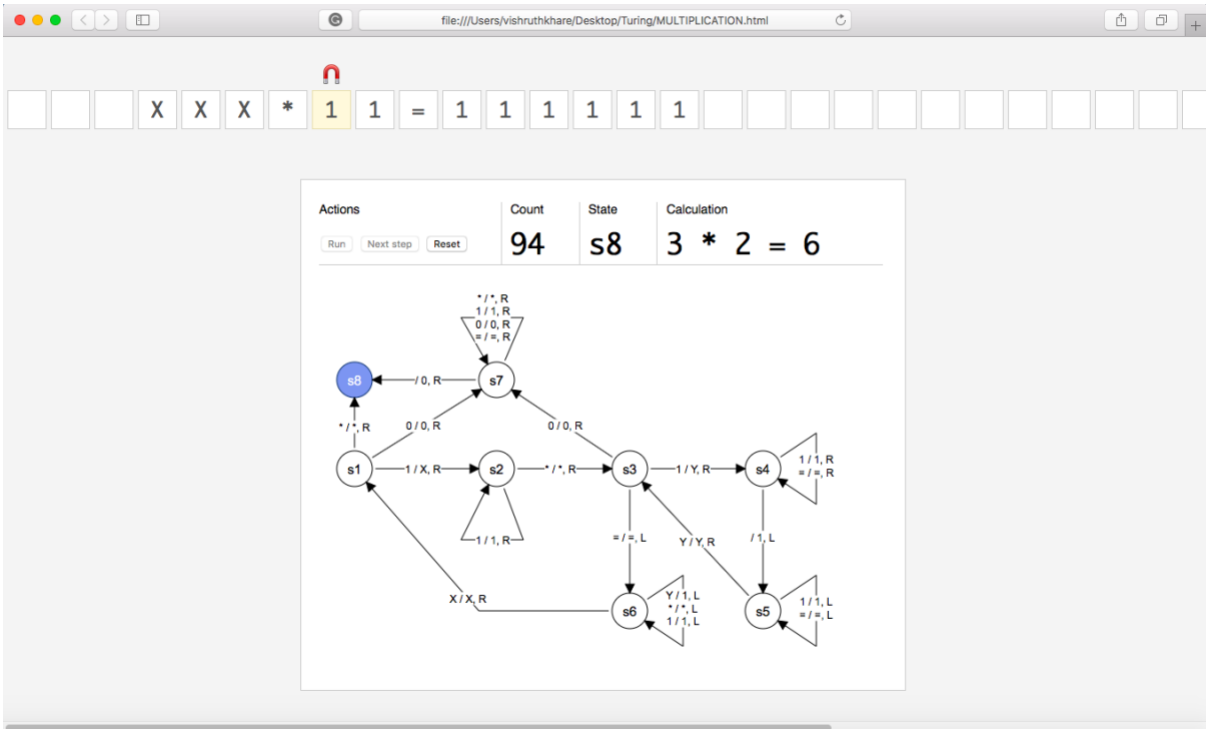
- First convert the initial 1 into X and go to the next state and keep looping until you find a ∗
- Once the machine encounters a ∗, it goes to the next state and changes the first 1 it sees into Y and go to the end until it encounters a Blank Space and inserts a 1 in that Blank space.
- Then come back until it encounters a Y, the state changes at this point and the machine iterates one step to the right.
- If it is a 1 then repeat the above step if it is = then go back and convert all the Ys to 1s.
- Then iterate to the left until you encounter a X then the state changes and we go to the right.
- If the encountered symbol is ∗, that means all the alphabets of the 1st number are over and hence the simulation has come to an end.
- If there is an additional 1 remaining, then mark that 1 into X and continue the simulation. The number of 1s in the final result gives up the answer.



(A simulation where the machine is at s5 state)



Turing machine after completion of 1∗2, we can see that the 1st 1 has been accounted for and the value has been marked as X and the corresponding values as Y



(Termination of Turing Machine)

So after completion of 94 Epochs, we have the final result. The project that we have developed is for variable inputs and hence the epoch may vary as we continue the simulation.

	INPUT						
STATES	1	*	=	X	Y	" "	0
S1	S2,X,R	S8,*,R					S7,0,R
S2	S2,1,R	S3,*,R					
S3	S4,Y,R		S6,=,L				S7,0,R
S4	S4,1,R		S4,=,R			S5,1,L	
S5	S5,1,L		S5,=,L		S3,Y,R		
S6	S6,1,L	S6,*,L		S1,X,R	S6,1,L		
S7		S7,*,R	S7,=,R			S8,0,R	S7,0,R
S8							

Transition Table for Turing Machine for Multiplication

FACTORIAL

Using Turing Machine

The idea of calculating the factorial is that to perform

a! = a*(a-1)*.....*1=?

we can perform this task recursively as

a * (a-1)!

(a * (a-1)) * (a-2)!

‘ ‘

‘ ‘

‘ ‘

(a * (a-1).....) * 1

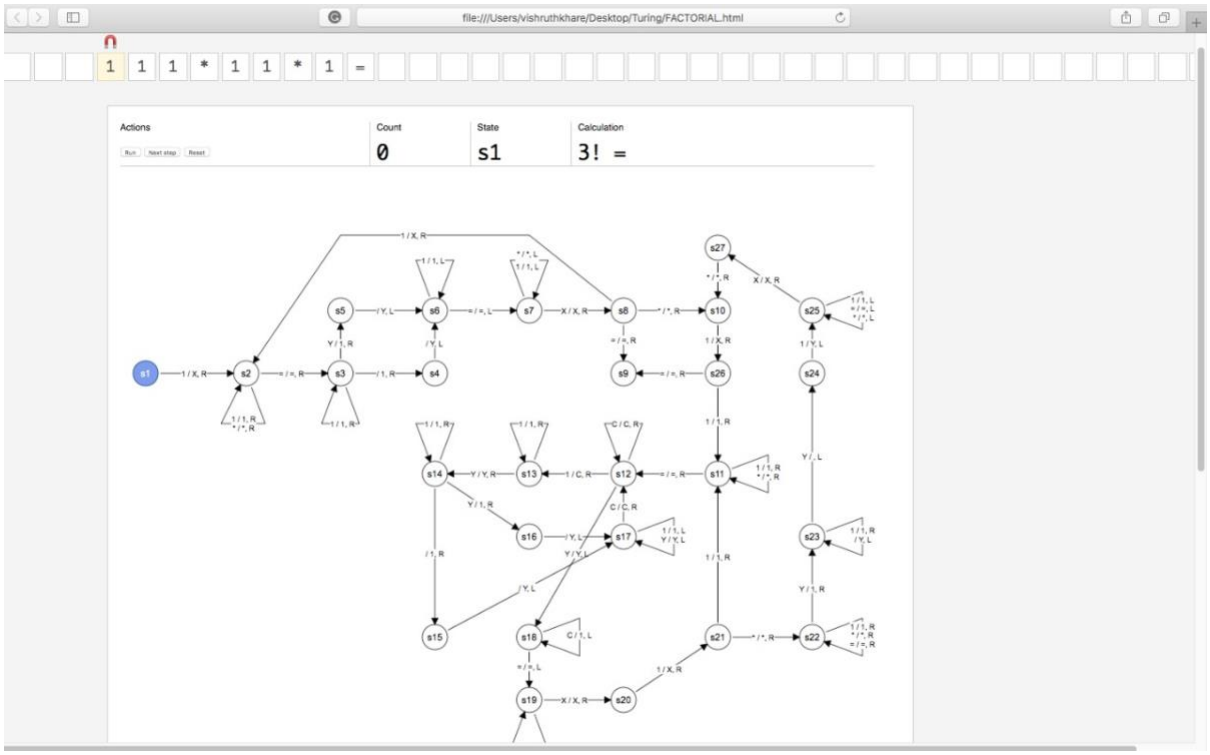
We are going to create a simulation that will multiply two numbers then take their product and multiply it with the next number until 1 is achieved. The multiplication procedure would done by a similar Turing machine as used above for multiplication. We have created a set of rules as to how the Turing Machine will respond to each state. The algorithm is as follows:

- First convert the initial 1 into X and go to the next state and keep looping until you find an =.
- When = is encountered the machine searches for either a blank space or the rightmost Y.
- If blank is encountered then 1 is inserted and then Y is inserted on the right of 1 to mark the ending
- If Y is encountered then Y is replace by 1 and then Y is inserted on the right of 1 to mark the ending
- Now the head moves leftward and loop until X is encountered
- Now the head moves one step right, if 1 is encountered then all the above steps are repeated, else if = is encountered head moves to the final state and the machine terminates.
- But if head encounters an * then the machine goes to a new state. Now the multiplication of two numbers will take place.
- Now upon converting the leftmost 1 to X, the machine loops until it finds an =.

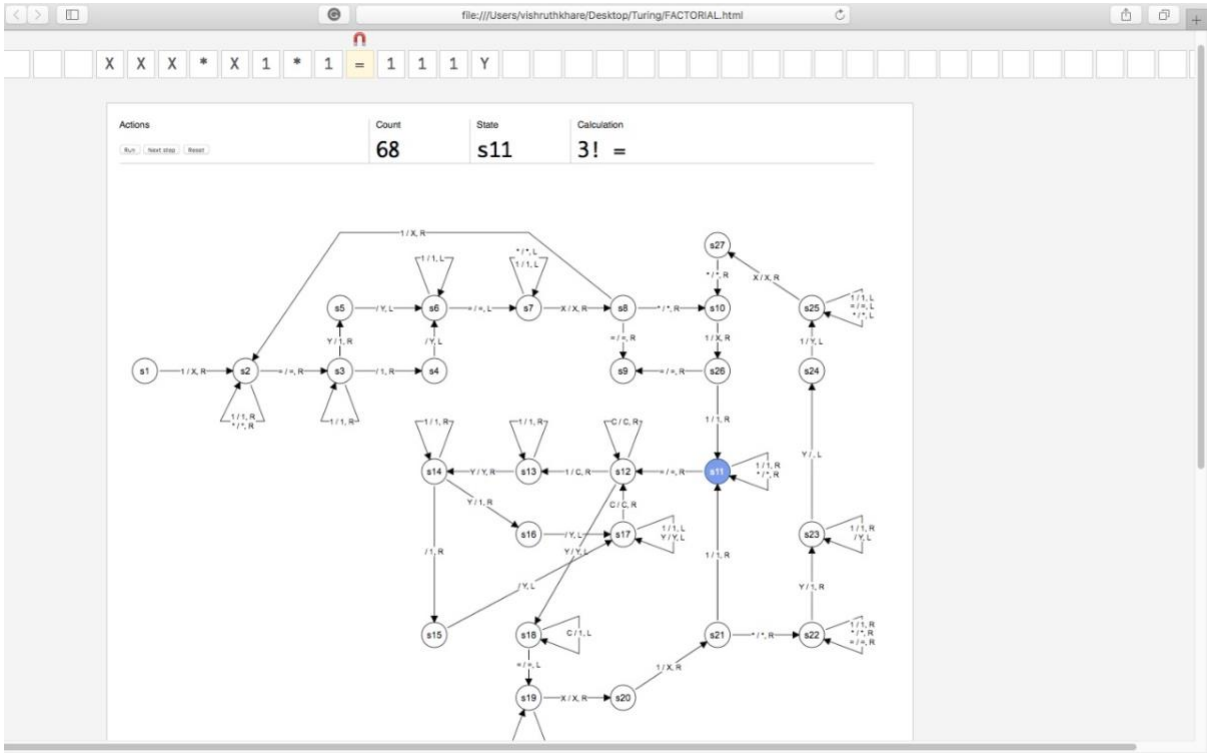
- Now upon the encountering 1 on the right of = 1 is converted to C and then 1 is inserted at the first blank space and then a Y to indicate the ending.
- Now each leftmost 1 after = and before Y are converted to C and the same number of 1 are inserted before the last Y.
- Now the Head moves leftward until it finds a C and then take one step rightward and if Y is encountered, covert all the C after = to 1.
- Now the head moves leftward until it encounters an X and then take one step right.
- Now head is on a 1 which is converted to X and then it takes one more step rightward, if head encounters 1 then above steps are repeated else if * is encountered then addition of the two numbers separated by Y takes place.
- Head converts the first Y it encounters after = to 1 then moves right and coverts the last Y to " " and then moves one step left and converts 1 to Y to indicate the ending
- Now, all the above steps are repeated until = is encountered just after a single 1.

Formal Logic for finding Factorial

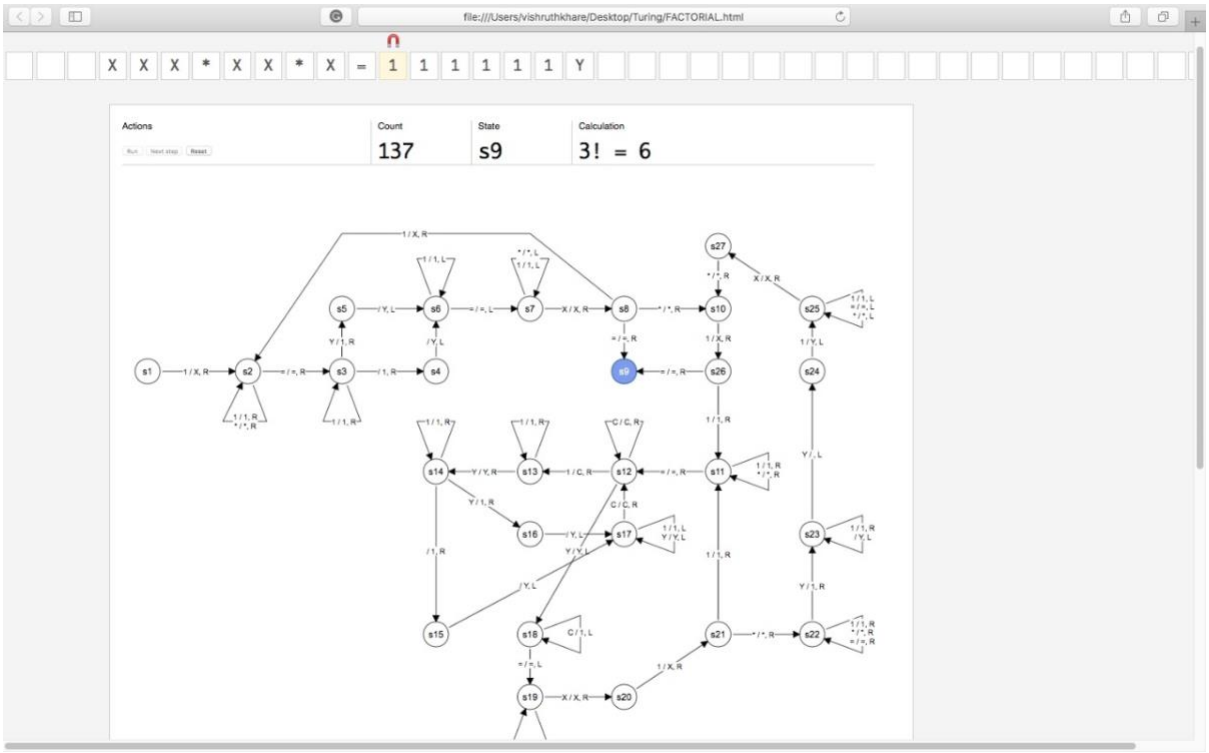
symbol	Interference
Q	S1,s2,…..,S26, S27
T	1,*,=,X,Y," ",C
B	" "
∑	1,*,=
F	S9
q0	S1



(Initial state of the machine just before the computation has begun)



(A simulation where the machine is at s11 state)



(Termination of Turing Machine)

So after completion of 137 Epochs, we have the final result for factorial of 3. The project that we have developed is for variable inputs and hence the epoch may vary as we continue the simulation.

INPUT

STATE	1	*	=	X	Y	" "	C
S1	S2,X,R						
S2	S2,1,R	S2,*,R	S3,=,R				
S3	S3,1,R				S5,1,R	S4,1,R	
S4						S6,Y,L	
S5						S6,Y,L	
S6	S6,1,L		S7,=,L				
S7	S7,1,L	S7,*,L		S8,X,R			
S8	S2,X,R	S10,*,R	S9,=,R				
S9							
S10	S26,X,R	S25,*,R					
S11	S11,1,R	S11,*,R	S12,=,R				
S12	S13,C,R				S18,Y,L		S12,C,R

S13	S13,1,R				S14,Y,R		
S14	S14,1,R				S16,1,R	S15,1,R	
S15						S17,Y,L	
S16						S17,Y,L	
S17	S17,1,L				S17,Y,L		S12,C,R
S18			S19,=,L				S18,1,L
S19	S19,1,L	S19,*,L		S20,X,R			
S20	S21,X,R						
S21	S11,1,R	S22,*,R					
S22	S22,1,R	S22,*,R	S22,=,R		S23,1,R		
S23	S23,1,R				S24, ,L		
S24	S25,Y,L						
S25	S25,1,L	S25,*,L	S25,=,L	S27,X,R			
S26	S11,1,R	S11,8,R	S9,=,R				
S27		S10,*,R					

Transition Table for Turing Machine for finding Factorial