LECTURE 5 – TURING MACHINES PART 1

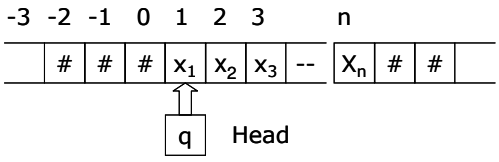
Overview

- The Turing Machine Model
- Examples of Turing Machine

The Turing Machine Model

The Turing machine (TM) is a simple and powerful mathematical model of computation that can be used as language acceptor, language translator, function evaluator. The languages that the Turing machine can compute are recursively enumerable.

The graphical model of the Turing machine is as follows:



A move in the Turing machine depends on the current input symbol (x_1 in the above illustration) and the current state (q in the above illustration).

A step in a Turing machine program is an action which is specified as a 5-tuple of the form:

(current state, current tape symbol, new state, new symbol, direction of move)

For example, the action (q,x_1,p,r,R) in the above illustration means that x_1 is replaced by r, q changes to p, and the head moves to the right and then points to the symbol x_2.

The formal model of the Turing machine is: TM = (Q , Σ , Γ , δ , q_0 , F) with:

Q is the set of states

Σ is the set of input symbols

Γ is the set of tape symbols including Σ and #.

δ is the transition (partial) function δ = Q x Γ → Q x Γ x { L , R }

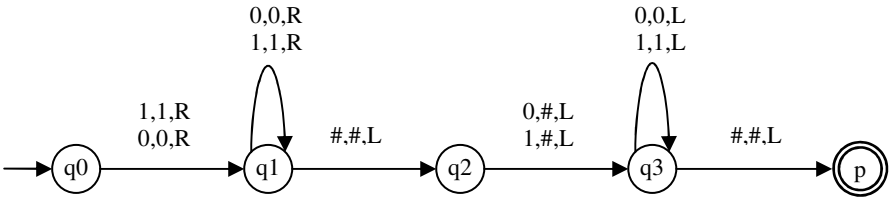
q_0 is the initial state

F is the set of final states

Turing machines can also be represented graphically using a state diagram which is a labelled directed graph.

Examples of Turing Machines

Simple Eraser. This Turing machine reads strings in the language given by the expression (0,1)* and replaces the right-most symbol by a blanc (#).



TM = ({q_0,q_1,q_2,q_3},{0,1},{0,1,#},δ,q_0,{p}) where δ is given by:

Σ / Q	0	1	#
q_0	(q_1 , 0 , R)	(q_1 , 1 , R)	∅
q_1	(q_1 , 0 , R)	(q_1 , 1 , R)	(q_2 , # , L)
q_2	(q_3 , # , L)	(q_3 , # , L)	∅
q_3	(q_3 , 0 , L)	(q_3 , 1 , L)	(p , # , L)

Then, the above Turing machine processes the input string "1110" as follows:

#$q_0$1110# → #1$q_1$110# → #11$q_1$10# → #111$q_1$0# → #1110q_1# → #111$q_2$0# → #11$q_3$1## → #1$q_3$11## → #$q_3$111## → q_3#111## → p##111##

Parity Counter. Design a Turing machine that reads binary strings and counts the number of 1's in the sequence. The output is 0 if the number of 1's in the string is even and 1 if the number of 1's is odd. The actions that should be carried out are:

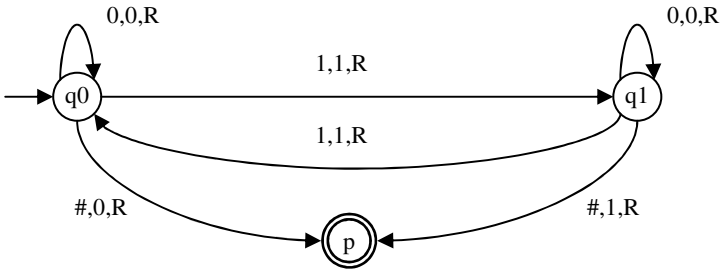
1. Search for 1's to the right.
2. The 0's in the sequence are read but ignored.
3. Maintain a count of whether the number of 1's is odd or even.
4. After a blanc (#) is found, write the output to the right of the input string.

The set of states is as follows:

q_0 searches to the right for a 1, indicates the number of 1's so far is even.

q_1 searches to the right for a 1, indicates the number of 1's so far is odd.

The Turing machine is represented by the following state diagram:



TM=($\{q_0,q_1\}$,$\{0,1\}$,$\{0,1,\#\}$,δ,q_0,$\{p\}$) where δ is given by:

Q \ Σ	0	1	#
q_0	(q_0 , 0 , R)	(q_1 , 1 , R)	(p , 0 , R)
q_1	(q_1 , 0 , R)	(q_0 , 1 , R)	(p , 1 , R)

Processing of the string "01#" is as follows:

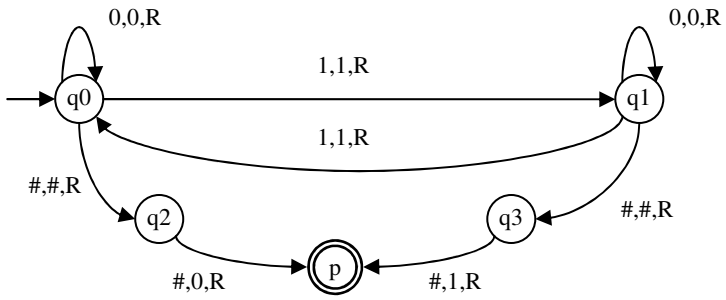
$q_001\# \rightarrow 0q_01\# \rightarrow 01q_1\# \rightarrow 011p$

Processing of the string "1011010#" is as follows:

$q_01011010\# \rightarrow 1q_1011010\# \rightarrow 10q_111010\# \rightarrow 101q_01010\# \rightarrow 1011q_1010\# \rightarrow 10110q_110\# \rightarrow 101101q_00\# \rightarrow 1011010q_0\# \rightarrow 10110100p$

Modify the above Turing machine so that the output is written to the right of the blanc symbol and then followed by another blanc symbol. That is, after processing the input string "01#" the tape should have the string "01#1#" and after processing "1011010#" the tape should have "1011010#0#"

The extra action that should be carried out is to search the blanc symbol and write the output to the right. The modified Turing machine is represented by the following state diagram:



Formally:

TM=($\{q_0,q_1,q_2,q_3,p\},\{0,1\},\{0,1,\#\},\delta,q_0,\{p\}$) where δ is given by:

Q \ Σ	0	1	#
q_0	(q_0 , 0 , R)	(q_1 , 1 , R)	(q_2 , # , R)
q_1	(q_1 , 0 , R)	(q_0 , 1 , R)	(q_3 , # , R)
q_2	∅	∅	(p , 0 , R)
q_3	∅	∅	(p , 1 , R)

Processing of the string "01#" is as follows:

$q_0 01\#\# \rightarrow 0q_0 1\# \rightarrow 01q_1\# \rightarrow 011\#q_3\# \rightarrow 011\#1p$

Processing of the string "1011010#" is as follows:

$q_0 1011010\# \rightarrow 1q_1 011010\# \rightarrow 10q_1 11010\# \rightarrow 101q_0 1010\# \rightarrow 1011q_1 010\# \rightarrow 10110q_1 10\# \rightarrow 101101q_0 0\# \rightarrow 1011010q_0\# \rightarrow 1011010\#q_2\# \rightarrow 1011010\#0p$

Unit Addition. Design a Turing machine that adds 1 to a number represented in binary possibly with leading zeros. The actions that should be carried out are:

1. Search to the right for the least significant bit.
2. Move to the left, add and carry. If a 0 is found replace it by 1 and halts. If a 1 is found replace it by 0 and continue to the left to take care of the carry.

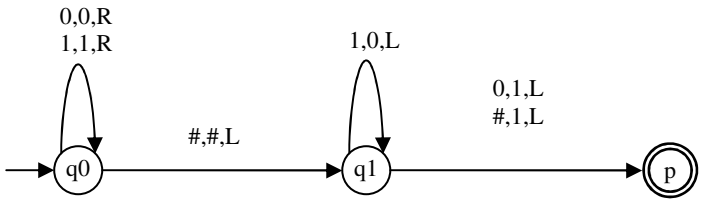
The set of states is as follows:

q_0 searches to the right for a the least significant bit.

q_1 moves to the left, if 0 found replaces it by 1 and halts, if 1 found replaces it by 0 and continues to the left.

TM=($\{q_0,q_1,p\},\{0,1\},\{0,1,\#\},\delta,q_0,\{p\}$) where δ is given by:

Σ / Q	**0**	**1**	**#**
q_0	(q_0 , 0 , R)	(q_0 , 1 , R)	(q_1 , # , L)
q_1	(p , 1 , L)	(q_1 , 0 , L)	(p , 1 , L)



Processing of the string "110#" is as follows:

$\#q_0110\# \rightarrow \#1q_010\# \rightarrow \#11q_00\# \rightarrow \#11q_00\# \rightarrow \#110q_0\# \rightarrow \#11q_10\# \rightarrow \#1p11\#$

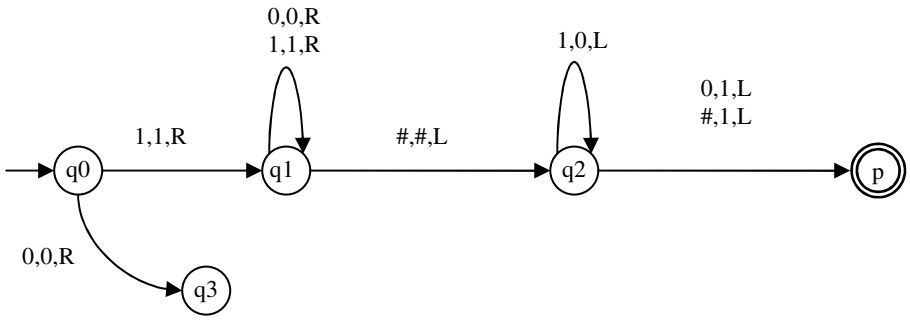
Processing of the string "11#" is as follows:

$\#q_011\# \rightarrow \#1q_01\# \rightarrow \#11q_0\# \rightarrow \#1q_11\# \rightarrow \#q_110\# \rightarrow \#q_1\#00\# \rightarrow p\#100\#$

Modify the above Turing machine so that the input string is computed only if it has no leading zeros.

The extra action is that the machine halts and fails if there are leading zeros.

Q \ Σ	0	1	#
q_0	(q_3 , 0 , R)	(q_1 , 1 , R)	∅
q_1	(q_1 , 0 , R)	(q_1 , 1 , R)	(q_2 , # , L)
q_2	(p , 1 , L)	(q_2 , 0 , L)	(p , 1 , L)



Parenthesis Checker. Design a Turing machine to recognise strings of matched parenthesis. The actions that should be carried out are:

1. Search for a right parenthesis.
2. Search for a matching left parenthesis.
3. If a right parenthesis is found then delete it and return to the matching left parenthesis (the one found in step 2) and delete it as well.
4. If there is not left parenthesis or there are too many right parenthesis, the Turing machine stops in a non-accepting state.

The set of states is as follows:

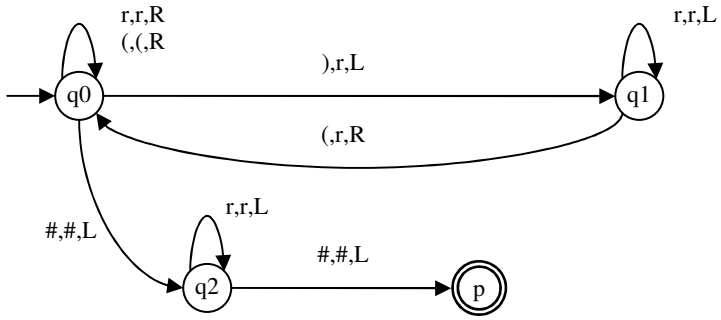
q_0 searches to the right for a right parenthesis and marks the position.

q_1 searches to the left for a left parenthesis that matches the right parenthesis already found and also marks the position. The machine stops if while in q_1 a blanc symbol is read because this means too many right parenthesis.

q_2 backwards search in order to verify that there are only marked positions. The machine stops if while in q_2 a no-blanc symbol is found and it accepts the input if a blanc is read because this means that the right side of the input string had been reached.

TM=($\{q_0,q_1,q_2,p\}$,{(,),r},{(,),#,r},δ,q_0,{p}) where δ is given by:

Q \ Σ	(	)	r	#
q_0	(q_0 , (, R)	(q_1 , r , L)	(q_0 , r , R)	(q_2 , # , L)
q_1	(q_0 , r , R)	∅	(q_1 , r , L)	∅
q_2	∅	∅	(q_2 , r , L)	(p , # , L)



Processing of the string "()(" is as follows:

$q_0()(\# \rightarrow (q_0)(\# \rightarrow q_1(r(\# \rightarrow rq_0r(\# \rightarrow rrq_0(\# \rightarrow rr(q_0\# \rightarrow rrq_2(\#$ → halts and fails

Therefore, the input string "()(" is not accepted by the Turing machine.

Processing of the string "()(())" is as follows:

$q_0()(())\# \rightarrow (q_0)(())\# \rightarrow q_1(r(()))\# \rightarrow rq_0r(())\# \rightarrow rrq_0(())\# \rightarrow rr(q_0())\# \rightarrow rr((q_0))\# \rightarrow rr(q_1(r)\# \rightarrow rr(rq_0r)\# \rightarrow rr(rrq_0)\# \rightarrow rr(rq_1rr\# \rightarrow rr(q_1rrr\# \rightarrow rrq_1(rrr\# \rightarrow rrrq_0rrr\# \rightarrow rrrrq_0rr\# \rightarrow rrrrrq_0r\# \rightarrow rrrrrrq_0\# \rightarrow rrrrrq_2r\# \rightarrow rrrrq_2rr\# \rightarrow rrrq_2rrr\# \rightarrow rrq_2rrrr\# \rightarrow rq_2rrrrr\# \rightarrow q_2rrrrrr\# \rightarrow q_2\#rrrrrr\# \rightarrow p\#\#rrrrrr\#$

Therefore, the input string "()(())" is accepted by the Turing machine.

Exercise 5.1 Design a Turing machine that reads binary strings and performs the following actions. If the input represents an odd number, subtracts one to the number. If the input represents an even number, add one to the number. For example, for input "101" the output should be "100" and for input "1010" the output should be "1011".

Exercise 5.2 Design a Turing machine that reads two unary numbers m and n separated by the symbol $ and computes m + n. Note that in unary representation a number is represented by the number of 1's in the sequence. That is, the requested Turing machine should read strings of the form m$n# (e.g. 1$11#) and produce the output mn# (e.g. 111#).

Reading: (Papadimitriou, chapter 2), (Boolos, chapter 3), (Lewis&Papadimitriou, chapter 4)