Turing Machines

KR Chowdhary
Professor & Head
Email: kr.chowdhary@ieee.org

Department of Computer Science and Engineering
MBM Engineering College, Jodhpur

October 25, 2010

Turing Machine: Agenda

- Alan M. Turing
- Church-Turing Thesis
- Definitions
- Computation
- TM Configurations

Alan M. Turing

- Alan Turing was one of the founding fathers of CS.
- His computer model - the Turing Machine - was inspiration/premonition of the electronic computer that came two decades later
- Was instrumental in cracking the Nazi Enigma cryptosystem in WW-II
- Invented the Turing Test used in AI
- Legacy: The Turing Award, eminent award in Theoretical CS research

Turing Thesis

- **Any algorithm can be carried out by Turing machine**

solving equations

- Consider Fermat's equation(last theorem): $x^n + y^n = z^n$, where n is a positive integer
- Question: Given an n, does this equation have a solution in integers?
- We could create a Turing machine T which would search for solutions when given an input n by working through a list of integers.
- Then $T(n)$ would halt when n=*1* (having found a solution, such as x=*1*, y=*1* and z=*2*, and when n=*2* (having found a solution such as x=*3*, y=*4* and z=*5*
- But T would not halt for any other input

Church's Thesis

- **Any reasonable attempt to model mathematically computer algorithms and their performance, is bound to end up with a model of computation and associated time cost, that is equivalent to Turing machines within a polynomial.**

A Thinking Machine: e.g., Successor Program

- Sample Rules:

  If read 1, write 0, Go Right, repeat.

  If read 0, write 1, HALT!
- If read #, write 1, HALT!
- Let's see how they are carried out on a piece of paper that contains the reverse binary representation of 47:red colored number represents position of head.

  1 1 1 1 0 1 #

  0 1 1 1 0 1 #

  0 0 1 1 0 1 #

  0 0 0 1 0 1 #

  0 0 0 0 0 1 #

  0 0 0 0 1 1 # HALTS; the result is reverse of 48.

A Thinking Machine: e.g., Successor Program

- So the successor's output on 111101 was 000011 which is the reverse binary representation of 48.
- Similarly, the successor of 127 should be 128.

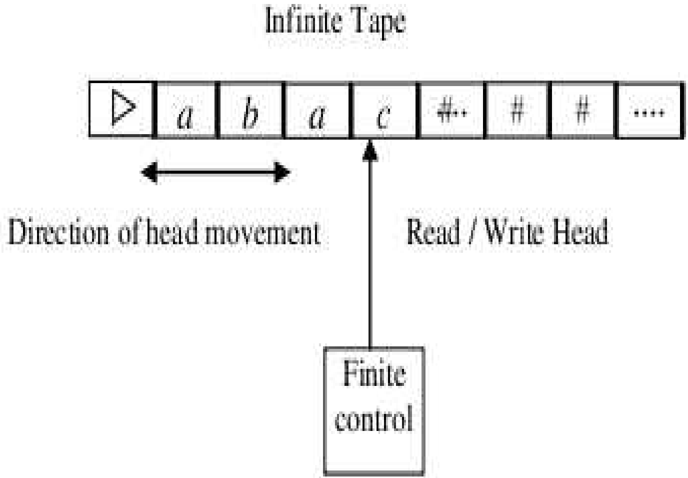
A Thinking Machine

- It was hard for the ancients to believe that any algorithm could be carried out on such a device. For us, it's much easier to believe, especially if you have programmed in assembly!
- However, ancients did finally believe Turing when Church's Lambda-calculus paradigm (on which lisp programming is based) proved equivalent!

Informal discussions

- A Turing machine (TM) is similar to a finite automaton with an unlimited and unrestricted memory. A Turing machine is however a more accurate model of a general purpose computer
- A Turing machine can do everything that a real computer can do
- But A Turing machine cannot solve certain classes of problems

Turing Machine Model for computation



Formal Model of Turing Machine

$$M = (Q, \Sigma, \mathit{\Gamma}, \delta, s, H)$$
$$\mathit{\Gamma} = \Sigma \cup \{\#, \triangleright\}$$
$$H \subseteq Q$$
$$\delta : (Q - H) \times \mathit{\Gamma} \rightarrow Q \times \mathit{\Gamma} \times \{L, R\}$$

Q is states

H is Halting states

Σ is set of input symbols

δ is transition function.

A Thinking Machine

- A Turing Machine (TM) is a device with a finite amount of read-only hard memory (states), and an unbounded amount of read/write tape-memory. There is no separate input. Rather, the input is assumed to reside on the tape at the time when the TM starts running.
- Just as with Automata, TM's can either be input/output machines (compare with Finite State Transducers), or yes/no decision machines.

Turing Machine

- 1936: Given a logical arithmetic computation, for which complete instructions for carrying out are supplied, it is possible to design a TM which can perform this computation
- TM v/s Human: states, memory, scratch pad paper
- TM v/s olden days computers
- TM v/s PDA
- $\{a^n b^n | n \geq 0\}$ v/s $\{a^n b^n c^n | n \geq 0\}$
- powers of TM

Turing Machine Criteria

- These are Automata
- As simple as possible - to define formally, describe and reason about them
- As general as possible (any computation can be represented by them)

Acceptability by Turing machine

A string w is accepted by M if after being put on the tape with the Turing machine head set to the left-most position, and letting M run, M eventually enters the halting state state. In this case w is an element of L(M),the language accepted by M:-

$$L(M) = \{w | w \in \Sigma^* \wedge q_0 w \Rightarrow^* y\}$$

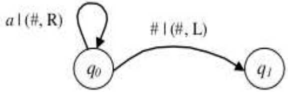
where, y is halting configuration

Turing Machine solves a Problem: Erase all a's

Consider a TM $M = (Q, \Sigma, \Gamma, \delta, s, H), Q = \{q_0, q_1\}$,
$\Sigma = \{a\}, \Gamma = \{a, \#, \triangleright\}, s = q_0$,
$\delta(q_0, a) = (q_0, \#, R)$
$\delta(q_0, \#) = (q_1, \#, L)$

- $w = aaaa$

  $q_0 aaaa\#$
  $\vdash \#q_0 aaa\#$
  $\vdash \#\#q_0 aa\#$
  $\vdash \#\#\#q_0 a\#$
  $\vdash \#\#\#\#q_0\#$
  $\vdash \#\#\#q_1\#$



Representation of a Configuration

If $i - 1 = n$ then $X_i = \#$. If $i = 1$ then $X_i = \triangleright$ and the head will move to right, else it will fall off the tape or we say it crashes. If $i > 1$ and $i = n$ then for d = L, we write a move as

$$X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n \vdash_M X_1 X_2 \ldots X_{i-2} p X_{i-1} Y X_{i+1} \ldots X_n.$$

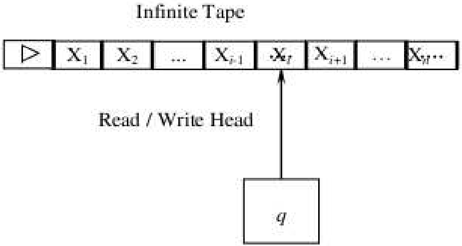


Figure 12.2: Turing Machine representing
ID = $X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n$

Alternatively, for $i > 1$ and d = R, a move is written as

$$X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n \vdash_M X_1 X_2 \ldots X_{i-1} Y p X_{i+1} \ldots X_n.$$

Configuration-1

- A configuration of a TM:
  - Current state
  - Symbols on tape
  - position of RW head
- A formal specification of configuration:
  - *uqv*, where *u*,*v* are strings on Σ, and *uv* is current content on tape, *q* is current state, and head is at first symbol of *v*.

    For example, $00101q_5011$ where read head points at *0* (third digit from end) and state is q_5.

Configuration-2

- For Two configurations:

$$uaq_i bv \;\; and \;\; uq_j acv, where, a, b, c \in \Sigma \;\; and \;\; u, v \in \Sigma^*$$

$$uaq_i bv \vdash uq_j acv \;\; if \;\; \delta(q_i, b) = (q_j, c, L)$$

$$uaq_i bv \vdash uacq_j v \;\; if \;\; \delta(q_i, b) = (q_j, c, R)$$

- Two special cases:

- The left most cell:

$$q_i bv \vdash q_j cv \;\; for \;\; \delta(q_i, b) = (q_j, c, L)$$

$$q_i bv \vdash cq_j v \;\; for \;\; \delta(q_i, b) = (q_j, c, R)$$

- On the cell with blank symbol:

$$uaq_i \;\; is \;\; equivalent \;\; to \;\; uaq_i\#$$

Example: of language recognition

- Design TM to accept: $a^n b^n, n \geq 1$
  1. let $M = (Q, \Sigma, \Gamma, \delta, s, H)$
  2. M replaces left most a by A, and then head moves to right until it encounters left most b
  3. Replaces this b by B, and then moves left to find the right most A. Then moves one step right to left most a
  4. Repeat Step 2 and 3 in order, i.e., 2, 3, 2, 3, ...
  5. When searching for b, if finds a blank character $\#$ (i.e., $|a^n| > |b^n|$), then M halts without accepting
  6. If a is not found but it finds b, then M halts without accepting, (i.e., $|a^n| < |b^n|$).
  7. After changing b to B, if M finds no more a then it checks that no more b remains. If this is true then $a^n b^n$ is accepted by M i.e., $|a^n| = |b^n|$)

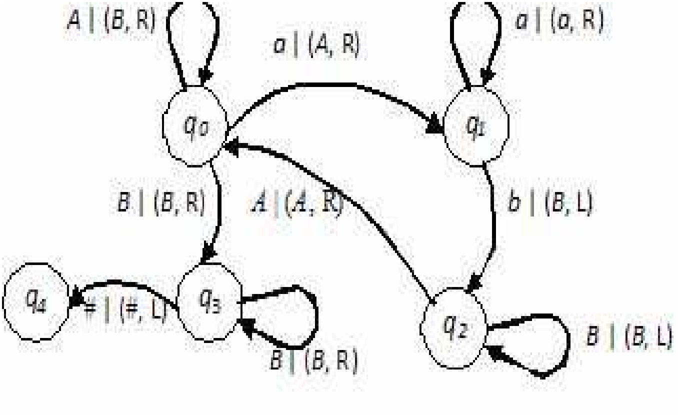
Example: of language recognition

$$Q = \{q_0, q_1, q_2, q_3, q_4\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{\triangleright, a, b, A, B, \#\}$$
$$s = q_0$$
$$H = \{q_4\}$$

Example: of language recognition...

- Design TM to accept: $a^n b^n, n \geq 1$
- Movement of RW head from start to right until b is found:
  $\delta(q_0, a) = (q_1, A, R)$
  $\delta(q_1, a) = (q_1, a, R), \delta(q_1, B) = (q_1, B, R)$
  $\delta(q_1, b) = (q_2, B, L)$
  $\delta(q_1, \#) = (q_1, \#, L)$, reject, when search for b fails
- move from R to L until A is found and start back:
  $\delta(q_2, B) = (q_2, B, L)$, traverse through B's
  $\delta(q_2, a) = (q_2, a, L)$, traverse a's
  $\delta(q_2, A) = (q_0, A, R)$, right most A is found
  $\delta(q_0, B) = (q_3, B, R)$, a's are exhausted
  $\delta(q_3, B) = (q_3, B, R)$, scan through B's
  $\delta(q_3, \#) = (q_4, \#, L)$, accept w when b's are over

Representation of a Configuration



Example: of language recognition Dry Run

- TM to accept: $a^n b^n, n \geq 1$
- Let $w = aabb \vdash q_0 aabb \vdash Aq_1 abb\#$
  $\vdash Aaq_1 bb\# \vdash Aaq_2 Bb\# \vdash Aq_2 aBb\#$
  $\vdash q_2 AaBb\# \vdash Aq_0 aBb\#$
  $\vdash AAq_1 Bb\# \vdash AABq_1 b\# \vdash AAq_2 BB\#$
  $\vdash Aq_2 ABB\# \vdash AAq_0 BB\#$
  $\vdash AABq_3 B\# \vdash AABBq_3 \# \vdash AABq_4 B\#$

Acceptors v/s deciders

- Let M is TM.
- Three possibilities occur on a given input w:
- M eventually enters q_{acc} and therefore halts and accepts. $w \in L(M)$
- M eventually enters q_{rej} or crashes somewhere. M rejects w, i.e., $w \notin L(M)$
- M never halts its computation and is caught up in an infinite loop. In this case w is neither accepted nor rejected. However, any string not explicitly accepted is considered to be outside the accepted language. $w \notin L(M)$
- decider: M never enters infinite loop.