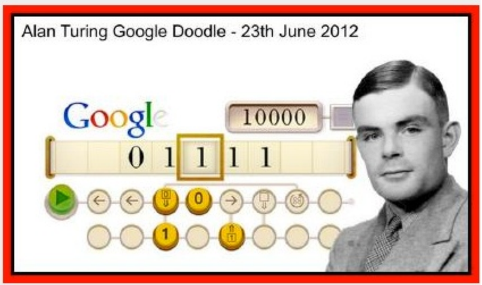
Turing Machines VI

Wednesday, October 29



Handouts:

Assignment 19

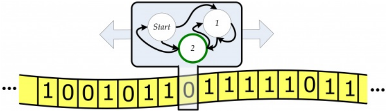
A Simple Turing Machine

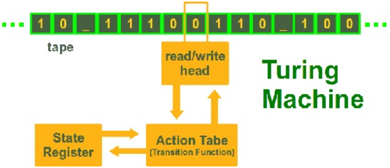
ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.


Start
1
2
... 1 0 0 1 0 1 1 0 1 1 1 1 1 0 1 1 ...


... 1 0 _ 1 1 1 0 0 1 1 0 _ 1 0 0 ...
tape
read/write head
Turing Machine
State Register
Action Tabe
(Transition Function)

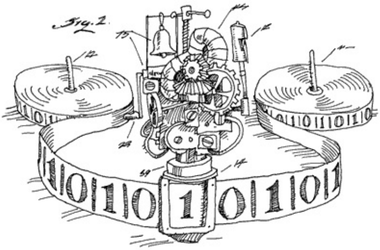
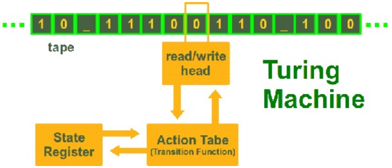


Fig. 2

What Exactly is a Turing Machine?



$$T = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

Where :

Q = Set of states

Σ = Input alphabet

Γ = Tape alphabet $(\Sigma \subseteq \Gamma)$

δ = Transition function

$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

q_0 = Start state

B = Blank symbol

F = Set of accepting states $(F \subseteq Q)$

Turing Machine Definition

1. a finite input alphabet Σ (used to create strings to be processed)
2. a finite tape alphabet Γ that contains Σ as well as a special blank symbol $\#$ (or $\square$ or $\clubsuit$) in Γ but not in Σ
3. a finite set of states **Q**
4. an infinite tape with cells (infinite to the left and right and indexed as ... $\#, \#$, -3, -2, -1, 0, 1,2 ,3, $\#, \# \; . \; . \; .$)
5. a tape head that can move left (**L**) or right (**R**) over the tape,reading and writing symbols onto tape cells as it proceeds
6. a special start state q_0
7. a special accept (or yes) state q_{yes} (or q_{acc})
8. a special reject (or no) state q_{no} (or q_{rej}) with $q_{acc} \neq q_{rej}$
9. a special transition function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ that describes how the **TM** operates on a specific input string.

Some Comments

- Everything is finite except for the tape. If you object to an infinite tape, you can simply view it as being unbounded (or unlimited).
- There are slight differences between how TMs can be defined but the definitions do not differ in significant ways.
- We shall view the special accept and reject states as final in the sense that once the TM enters one of these "halting" states the machine stops.
- We shall assume the tape cells are indexed by the integers . . . #, #, -3, -2, -1, 0, 1, 2 ,3 ,4, #, #, . . .
- And that each of the cells contains the special blank symbol just before we place any input string onto the tape.
- We will always place an input string to be "processed" onto the tape with its left end at cell 0.

So, for example, if the input string is *abcd* the contents of the tape will be

$$\{...\#, \#, \#, a, b, c, d, \#, \#, ...\}$$

(with a at cell 0).
Note that, given this assumption, we are guaranteed to have a blank symbol at each end of the input string.

More Comments

- The TM machine's tape head always starts at cell 0.
- The TM machine always starts in state q_0.
- This is often called the start configuration.
- It is the transition function $\delta()$ that determines the *action* of a specific TM.
  How do you read $\delta(q, a) = (q', b, L)$
  given the definition $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$?
  Answer: when in state q scanning symbol a on the tape, write b over the tape cell, move the head left by one cell, and enter state q'.
- In terms of modern computers, you can think of the tape as being (an unlimited amount of) *memory* and the transition function (along with the states) as being the *control unit* (within the CPU).

How it Works

A Turing machine begins in its start state with its tape head at cell 0 scanning the leftmost symbol in the string. At each step, it reads the symbol under its head, and, depending on that symbol and its current state, it

- writes a new symbol on that cell (possibly the same),
- moves its head either left or right one cell, and
- enters a (possibly) new state.

The action is determined by the **transition function** $\delta(q, a) = (q', b, L)$ given as part of the description for the machine.

A TM **accepts** its input (saying **yes**) by entering the accept state and **rejects** its input (saying **no)** by entering the reject state.

A TM to compute $x + y$ where x, y are positive integers

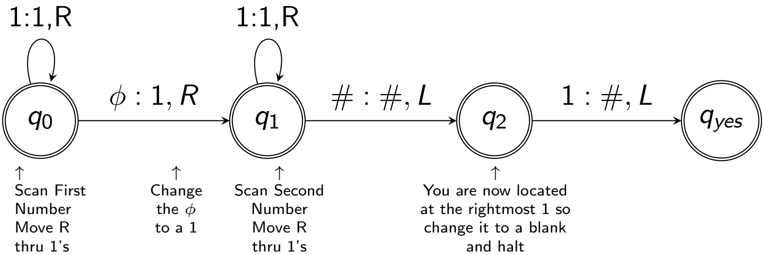
We use unary (|||||||) to represent x, y and $x + y/$
The input string will be $x\phi y$ in unary.
The goal is to produce xy on the tape.

Example: $3 + 5 = 8$

Start	. . .	#	1	1	1	ϕ	1	1	1	1	1	# . . .
				$\uparrow$ q_0								
Finish	. . .	#	1	1	1	1	1	1	1	1	#	. . .
										$\uparrow$ q_{yes}		

$\Sigma = \{\phi, 1\}$
$\Gamma = \{\phi, 1, \#\}$
Aside: We do not really need ϕ since we could just as easily use $\#$ to separate two numbers.

State Diagram



In our Transition Table, we will Reject on anything else (not drawn in the State Diagram)

You can use one fewer state since q_0 and q_1 really do the same thing.

Transition Table

Tape Symbols

		1	ϕ	$\#$	
s	q_0	$q_0, 1, R$	$q_1, 1, R$	$q_N, \#, L$	
t	q_1	$q_1, 1, R$	$q_N, -, -$	$q_2, \#, L$	$(q', sym, R/L)$
a	q_2	$q_Y, \#, L$	$q_N, -, -$	$q_N, -, -$	
t	q_Y	$-$	$-$	$-$	
e	q_N	$-$	$-$	$-$	

Another Turing Machine Transition Table

Here $\Sigma = \{1, *\}$ and $\Gamma = \{1, *, \#\}$

	1	#
q_0	$q_1, *, R$	$q_{yes}, *, R$
q_1	$q_0, *, R$	$q_{yes}, 1, R$
q_{yes}	—	—

The input tape will consists of some #'s followed by some nonnegative number of 1's followed by all #'s.

Input (a)	...	#	#	1	1	#	#	...

Output? Machine halts with $---\#\# *** \#\#...$ on the tape.

Input (a)	...	#	#	1	1	1	#	#	...

Output? Machine halts with $---\#\# *** 1\#\#...$ on the tape.
TM determines whether the number is Odd or Even. If Odd, TM prints a single 1 but if the number is Even, it prints all #'s.

On some input, a TM may run indefinitely without ever accepting or rejecting.

For example, it may **loop** on that input. Such a machine will never **halt**.

Simple Examples

1. A TM to *recognize* 2 (a *decider*)
2. A TM to *compute* $f(x) = x + 1$ for an integer $x \geq 1$ (the *increment* or *plus one* function).
3. A TM that does not halt (several, actually).
4. A TM to 'compute' $f(x) = 2x$ for an integer $x \geq 1$ (the *doubling function*)
5. A TM that adds two non-negative integers:

   $f(x, y) = x + y$ for two non-negative integers x and y.

Note: Leavitt discusses a TM to add two integers (page 68) but since he never gives a careful, precise definition for a TM, it is difficult to follow some of what he says.
Does his TM only work for $2 + 2$?.