Turing Machines

Prof. Philip Pennance[1] -Version: January 16, 2017

1. Let f be a function. Informally, a procedure for the evaluation of f is called "effective" or "mechanical" if it can be expressed by a finite list of finite instructions which together specify a finite number of paper and pencil operations which without the help of human ingenuity produce the value of $f(a)$ for each a.

2. Example: Let $\mathcal{F}$ be the set of formulae of CP the propositional calculus and let $\chi_T : \mathcal{F} \to \{0, 1\}$ be the characteristic function on the set T of provable formulae defined by:

$$\chi_T(F) = \begin{cases} 1 & \text{if } \vdash F, \\ 0 & \text{if } \neg \vdash F. \end{cases}$$

   Let $F \in \mathcal{F}$. It is a theorem of CP that if F is semantically true, then it is also provable i.e.

$$\models F \quad \Rightarrow \quad \vdash F$$

   It follows that the truth table procedure is an "effective" procedure for evaluating χ_T or equivalently for solving the decision problem "is F provable?".

3. Remark. The problem in (2) above, in the case of an arbitrary formal system, is known as Hilbert's *Entscheidungsproblem* "find an effective procedure which decides for given statement F whether or not it is logically valid (i.e. whether or not $\vdash F$).

4. The rough definition of "effective computability" will now made precise by appropriately defining the notion of "algorithm" using the the notion of a deterministic Turing machine DTM. The description of such a machine will guarantees that any "Turing computable" function is "effectively" computable in the rough sense discussed above. Conversly, Turing's "thesis" assumes that every effectively computable function is Turing computable. In this way, we identify the class of of effectivly computable with Turing computable functions.

5. Turing's thesis is not a theorem. It cannot be proven since the rough notion of effectively computable function as defined in item 1 is not a mathematical notion but an assumption based upon human experience. However, the identification of effective computability with Turing computability permits a mathematically precise definition of effective procedure or algorithm.

6. Another approach due to Church expresses this thesis in terms of recursive functions: Effectively computable functions are recursive.

7. A *deterministic Turing machines* (DTM) consist of an infinite tape and a read write head as depicted below. The positions on the tape are called cells and numbered as shown.

$\cdots$	-2	-1	1	2	3	$\cdots$		
$\cdots$	b	b	1	0	1	0	0	b $\cdots$
			$\uparrow$					
			$\boxed{q_0}$					

8. A *program* for a DTM is a quadruple $(Q, \Sigma, \Gamma, \delta)$ where

[1] http://pennance.us

(a) $Q = \{q_0, q_1, \ldots, q_m, q_Y, q_N\}$ is a finite alphabet of elements called *head states*. q_0 is called the *start state*. q_Y and q_N are the *final yes* and *final no* states.

(b) Γ is a finite set whose elements are called *tape symbols*, one of which denoted b is called *blank*

(c) Σ is a subset of $\Gamma \setminus b$. The elements of Σ are called input symbols.

(d) δ is a function

$$\delta : Q\backslash\{q_Y, q_N\}\times\Gamma \to Q\times\Gamma\times\{+1, -1\}$$

called the *transition function*.

9. The mode of operation of such a DTM is as follows:

   (a) Initialization

      i. The head position is initialized to cell 1

      ii. The head state q set to q_0

      iii. An input word $w \in \Sigma^*$ is placed in cells $1, 2, \ldots, |w|$. one symbol per cell.

      iv. The blank symbol b is placed in all other cells.

   (b) Iteration: If $q \notin \{q_Y, q_N\}$ is current head state and σ is value in current cell and $\delta(q, \sigma) = (q', \sigma', d)$ then

      i. Head writes σ' in place of σ

      ii. Head state is changed from q to q'.

      iii. Head moves left if $d = -1$ or right if $d = +1$.

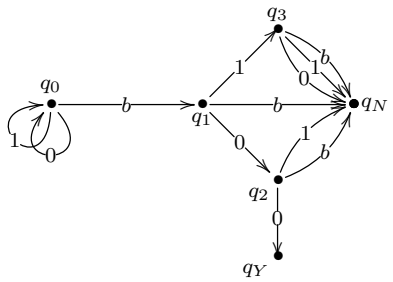
      iv. If $q' \in \{q_Y, q_N\}$ stop.

10. Since δ is a function, the operation of the DTM is completely determined by the graph of δ i.e. by a list of quintuples $(q, \sigma, q', \sigma', d)$. (Notice that only one quintuple can begin with a given pair (q, σ).)

11. Example: Let
$Q = \{q_0, q_1, q_2, q_3, q_y, q_N\}$
$\Gamma = \{0, 1, b\}$; $\Sigma = \{0, 1\}$
Define a transition function δ by the table:

	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

12. The following digraph shows the effect of the tape symbol on each head state.



The vertex set V is the set of head states, and for each tape symbol γ an edge qq' is drawn where q' is the unique head state satisfying $\delta(q, \gamma) = (q', \gamma', d)$ for some γ'.

13. The operation of the DTM is depicted below for input string 10100. It is simple to check that for this input the machine stops on q_Y.

(a)

$\cdots$	b	b	1	0	1	0	0	b	$\cdots$
			↑						
			$\boxed{q_0}$						

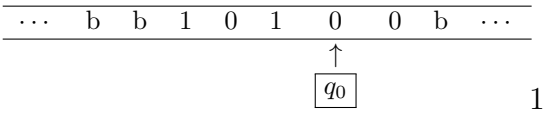
(b)

$\cdots$	b	b	1	0	1	0	0	b	$\cdots$
				↑					
				$\boxed{q_0}$					

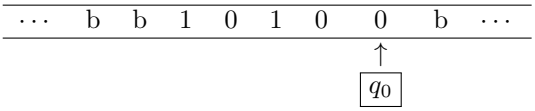
(c)

$\cdots$	b	b	1	0	1	0	0	b	$\cdots$
					↑				
					$\boxed{q_0}$				

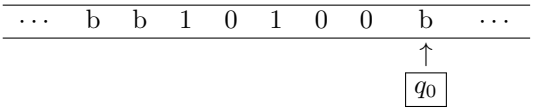
(d)



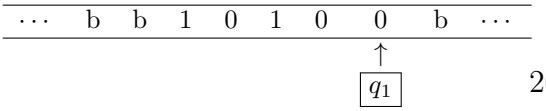
(e)



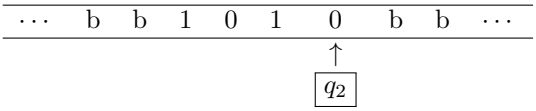
(f)



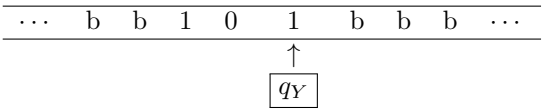
(g)



(h)



(i)



14. A DTM program M is called an *algorithm* if M halts (either in q_Y or q_N) for all input strings in Σ^*

15. Example: The DTM program in (11) above is an algorithm.
It is easy to see that:

    (a) The tape head initially moves to the right without writing to the tape or changing state until a blank is encountered.

    (b) The head state then changes to q_1. from which it can never return to q_0. Henceforth the head moves left and writes only blanks.

    (c) If the input string ends in 00 then the program will stop on head-state q_Y. Otherwise it stops on q_N.

16. We say that a DTM program M *accepts* a string $x \in \Sigma^*$ if M halts in state q_y when given input x.

17. The set

$$L_M = \{x \in \Sigma^* : M \text{ accepts } x\}$$

is called the language *recognized by* M.

18. Example: The language recognized by M in (11) above is

$$L_M = \{x \in \{0,1\}^* : x \text{ terminates in } 00\}$$

19. A decision problem (roughly speaking, is a problem with a yes-no solution.

20. Example
Problem Π: Integer Divisibility by Four
Instance: $n \in \mathbb{N}$
Question: $\exists m \in \mathbb{Z}$ such that $n = 4m$?

21. Each instance n of the problem Pi can be encoded as a word in the universal language $\{0,1\}^*$ by the function $e : \mathbb{N} \to \{0,1\}^*$ given letting $e(n)$ be the binary representation of n. Now binary number is divisible by 4 if and only if it terminates in 00. Thus the "Yes" instances of Pi are precisely those binary numbers termination in 00.

22. More generally, let Π is a problem with encoding e in alphabet Σ. Then the set $L((\Pi, e)$ of words $\sigma \in \Sigma^*$ which encode a yes instances of Π is called the *language determined by the decision problem* Π *under the encoding* e

23. A DTM program M is said to *solve* a decision problem Π if

    (a) M is an algorithm (i.e. halts for any input string)

    (b) $L_M = L((\Pi, e)$

24. Example: The discussion shows that the algorithm M given in (11) solves the problem Π: Integer divisibility by 4.

25. Notice that decision type problems can often be reduced to the evaluation of a corresponding "problem" function $f_\Pi : D_\pi \to \{0,1\}$. The problem of divisibility by 4 has problem function $f : \mathbb{N} \to \{0,1\}$ is given by:

$$f_\pi(n) = \begin{cases} 1 & \text{if 4 divides } n, \\ 0 & \text{otherwise.} \end{cases}$$

This function is Turing computable in the sense that the algorithm M can easily be modified to ouput the value of f for any encoding of an instance $n \in \mathbb{N}$ of Π.

26. Example. Let Σ be an arbitrary alphabet and $L \subseteq \Sigma^*$ a language in L. Given a word $\sigma \in \Sigma^*$ the problem of determining whether or not $\sigma \in L$ is called the *recognition problem* RP for L. From our discussion, a solution of RP is an algorithm M which accepts L, i.e.

    (a) M stops for all $\sigma \in \Sigma^*$

    (b) M stops on q_Y if and only if $\sigma \in L$.