Design and Analysis of Algorithms

Turing Machine

Dr. Soharab Hossain Shaikh
BML Munjal University

1

Turing Machine

Computability and Decidability

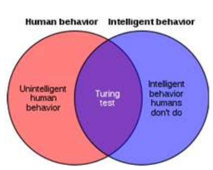
2

Alan Turing

- Alan Turing characterized computable functions by building a machine. Though theoretical this gave rise to the idea of computers.
- But Turing also worked on ideas and concepts that later made profound impact in AI.

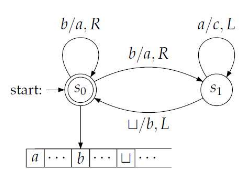
The *Enigma* machine is a cipher device developed and used in the early- to mid-20th century to protect commercial, diplomatic, and military communication. It was employed extensively by Nazi Germany during World War II, in all branches of the German military.

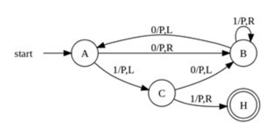
Cracking the *Enigma* code



3

Turing Machine





4

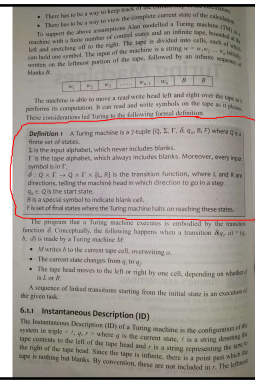
Turing Machine

A quick review of Turing Machine
before discussion on the
Halting Problem

Content taken from the book - ***Formal Languages and Automata Theory by KVN Sunitha & N Kalyani, McGrawHill***

5

Definition of a
Turing Machine



- There has to be a way to keep track of the current step of the calculation.
- There has to be a way to view the complete current state of the calculation.

To support the above assumptions Alan modelled a Turing machine (TM) as a machine with a finite number of control states and an infinite tape, bounded at the left and stretching off to the right. The tape is divided into cells, each of which can hold one symbol. The input of the machine is a string $w = w_1 w_2 \ldots w_n$ initially written on the leftmost portion of the tape, followed by an infinite sequence of blanks B.

w_1	w_2	w_3	......	w_{n-1}	w_n	B	B

The machine is able to move a read/write head left and right over the tape as it performs its computation. It can read and write symbols on the tape as it pleases. These considerations led Turing to the following formal definition.

Definition 1 A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where Q is a finite set of states.
Σ is the input alphabet, which never includes blanks.
Γ is the tape alphabet, which always includes blanks. Moreover, every input symbol is in Γ.
$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function, where L and R are directions, telling the machine head in which direction to go in a step.
$q_0 \in Q$ is the start state.
B is a special symbol to indicate blank cell.
F is set of final states where the Turing machine halts on reaching these states.

The program that a Turing machine executes is embodied by the transition function δ. Conceptually, the following happens when a transition $\delta(q_i, a) = (q_j, b, d)$ is made by a Turing machine M:

- M writes b to the current tape cell, overwriting a.
- The current state changes from q_i to q_j.
- The tape head moves to the left or right by one cell, depending on whether d is L or R.

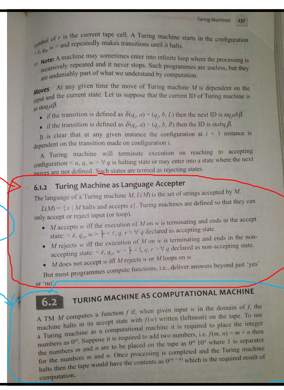
A sequence of linked transitions starting from the initial state is an execution of the given task.

6.1.1 Instantaneous Description (ID)

The Instantaneous Description (ID) of a Turing machine is the configuration of the system in triple $< \ell, q, r >$ where q is the current state, ℓ is a string denoting the tape contents to the left of the tape head and r is a string representing the tape to the right of the tape head. Since the tape is infinite, there is a point past which the tape is nothing but blanks. By convention, these are not included in r. The leftmost

6

Turing Machine as
- Language Acceptor
- Computational Machine



Turing Machines 237

symbol of r is the current tape cell. A Turing machine starts in the configuration $< \varepsilon, q_0, w >$ and repeatedly makes transitions until it halts.

Note: A machine may sometimes enter into infinite loop where the processing is recursively repeated and it never stops. Such programmes are useless, but they are undeniably part of what we understand by computation.

Moves At any given time the move of Turing machine M is dependent on the input and the current state. Let us suppose that the current ID of Turing machine is as $\alpha a q_i a \beta$.

- if the transition is defined as $\delta(q_i, a) = (q_j, b, L)$ then the next ID is $\alpha q_j a b \beta$.
- if the transition is defined as $\delta(q_i, a) = (q_j, b, R)$ then the ID is $\alpha a b q_j \beta$.

It is clear that at any given instance the configuration at $i + 1$ instance is dependent on the transition made on configuration i.

A Turing machine will terminate execution on reaching to accepting configuration $< u, q, w > \forall q$ is halting state or may enter into a state where the next moves are not defined. Such states are termed as rejecting states.

6.1.2 Turing Machine as Language Accepter

The language of a Turing machine M, $L(M)$ is the set of strings accepted by M.
$L(M) = \{x \mid M \text{ halts and accepts } x\}$. Turing machines are defined so that they can only accept or reject input (or loop).

- M accepts w iff the execution of M on w is terminating and ends in the accept state: $< \varepsilon, q_o, w > \vdash < \ell, q, r > \forall q$ declared as accepting state.
- M rejects w iff the execution of M on w is terminating and ends in the non-accepting state: $< \varepsilon, q_o, w > \vdash < l, q, r > \forall q$ declared as non-accepting state.
- M does not accept w iff M rejects w or M loops on w.

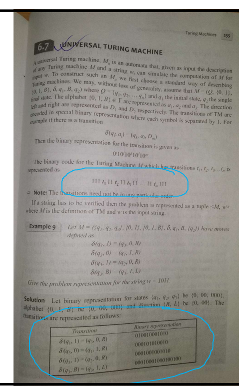
But most programmes compute functions, i.e., deliver answers beyond just 'yes' or 'no'.

6.2 TURING MACHINE AS COMPUTATIONAL MACHINE

A TM M computes a function f if, when given input w in the domain of f, the machine halts in its accept state with $f(w)$ written (leftmost) on the tape. To use a Turing machine as a computational machine it is required to place the integer numbers as 0^m. Suppose it is required to add two numbers, i.e. $f(m, n) = m + n$ then the numbers m and n are to be placed on the tape as $0^m 10^n$ where 1 is separator for the numbers m and n. Once processing is completed and the Turing machine halts then the tape would have the contents as $0^{(m+n)}$ which is the required result of computation.

7

Universal
Turing Machine



Turing Machines 255

6.7 UNIVERSAL TURING MACHINE

A universal Turing machine, M_u is an automata that, given as input the description of any Turing machine M and a string w, can simulate the computation of M for input w. To construct such an M_u we first choose a standard way of describing Turing machines. We may, without loss of generality, assume that $M = (Q, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, q_2)$ where $Q = \{q_1, q_2, \ldots q_n\}$ and q_1 the initial state, q_2 the single final state. The alphabet $\{0, 1, B\} \in \Gamma$ are represented as a_1, a_2 and a_3. The direction left and right are represented as D_1 and D_2 respectively. The transitions of TM are encoded in special binary representation where each symbol is separated by 1. For example if there is a transition

$$\delta(q_i, a_j) = (q_k, a_l, D_m)$$

Then the binary representation for the transition is given as

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

The binary code for the Turing Machine M which has transitions $t_1, t_2, t_3 \ldots t_n$ is represented as

$$111\ t_1\ 11\ t_2\ 11\ t_3\ 11 \ldots 11\ t_n\ 111$$

Note: The transitions need not be in any particular order.

If a string has to be verified then the problem is represented as a tuple $<M, w>$ where M is the definition of TM and w is the input string.

Example 9 *Let $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$ have moves defined as*

$$\delta(q_1, 1) = (q_3, 0, R)$$
$$\delta(q_3, 0) = (q_1, 1, R)$$
$$\delta(q_3, 1) = (q_2, 0, R)$$
$$\delta(q_3, B) = (q_3, 1, L)$$

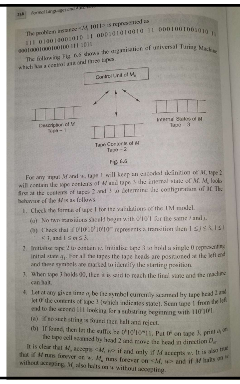
Give the problem representation for the string $w = 1011$.

Solution Let binary representation for states $\{q_1, q_2, q_3\}$ be $\{0, 00, 000\}$, alphabet $\{0, 1, B\}$ be $\{0, 00, 000\}$ and direction $\{R, L\}$ be $\{0, 00\}$. The transitions are represented as follows:

Transition	*Binary representation*
$\delta(q_1, 1) = (q_3, 0, R)$	010010001010
$\delta(q_3, 0) = (q_1, 1, R)$	000101010010
$\delta(q_3, 1) = (q_2, 0, R)$	0001001001010
$\delta(q_3, B) = (q_3, 1, L)$	00010001000100100

8

Representation of a Universal Turing Machine



256 *Formal Languages and Automata*

The problem instance <M, 1011> is represented as
111 010010001010 11 000101010010 11 0001001001010 11
0001000100010010 0 111 1011

The following Fig. 6.6 shows the organisation of universal Turing Machine which has a control unit and three tapes.

Fig. 6.6

For any input M and w, tape 1 will keep an encoded definition of M, tape 2 will contain the tape contents of M and tape 3 the internal state of M. M_u looks first at the contents of tapes 2 and 3 to determine the configuration of M. The behavior of the M is as follows.

1. Check the format of tape 1 for the validations of the TM model.
   (a) No two transitions should begin with 0^i10^j1 for the same i and j.
   (b) Check that if $0^i10^j10^k10^l10^m$ represents a transition then $1 \le j \le 3$, $1 \le l \le 3$, and $1 \le m \le 3$.
2. Initialise tape 2 to contain w. Initialise tape 3 to hold a single 0 representing initial state q_1. For all the tapes the tape heads are positioned at the left end and these symbols are marked to identify the starting position.
3. When tape 3 holds 00, then it is said to reach the final state and the machine can halt.
4. Let at any given time a_j be the symbol currently scanned by tape head 2 and let 0^i the contents of tape 3 (which indicates state). Scan tape 1 from the left end to the second 111 looking for a substring beginning with 110^i10^j1.
   (a) if no such string is found then halt and reject.
   (b) If found, then let the suffix be $0^k10^l10^m11$. Put 0^k on tape 3, print a_l on the tape cell scanned by head 2 and move the head in direction D_m.

It is clear that M_u accepts <M, w> if and only if M accepts w. It is also true that if M runs forever on w, M_u runs forever on <M, w> and if M halts on w without accepting, M_u also halts on w without accepting.

9

Turing Machine & Decidable Language

– A TM accepts language L if it has an accepting run on each word in L.
– A TM decides language L if it accepts L and halts on all inputs.

Decidable and Turing recognizable languages

– A language L is decidable (recursive) if there exists a Turing machine M which decides L (i.e., M halts on all inputs and M accepts L).

– A language L is Turing recognizable (recursively enumerable) if there exists a Turing machine M which accepts L.

10

Algorithms and Decidability

Algorithms $\Longleftrightarrow$ Decidable (i.e, TM decides it)

– A decision problem P is said to be decidable (i.e., have an algorithm) if the language L of all *yes* instances to P is decidable.

– A decision problem P is said to be semi-decidable (i.e., have a semi-algorithm) if the language L of all *yes* instances to P is r.e.

– A decision problem P is said to be undecidable if the language L of all *yes* instances to P is not decidable.

11

Church Turing Thesis

Lambda Calculus

– In 1936, Church introduced Lambda Calculus as a formal description of all computable functions.
– Independently, Turing had introduced his A-machines in 1936 too.
– Turing also showed that his A-machines were equivalent to Lambda Calculus of Church.

– So, can a Turing machine do everything? In other words are there algorithms to solve every question.

– If there is TM solving a problem, does there exist an equivalent TM that halts?

12

More on Church Turing Thesis

Lambda Calculus

- In 1936, Church introduced Lambda Calculus as a formal description of all computable functions.
- Independently, Turing had introduced his A-machines in 1936 too.
- Turing also showed that his A-machines were equivalent to Lambda Calculus of Church.

- So, can a Turing machine do everything? In other words are there algorithms to solve every question.

- If there is TM solving a problem, does there exist an equivalent TM that halts?

13

The Acceptance Problem for Turing Machine

Given a TM, does it accept a given input word?

$L^A_{TM} = \{< M, w > | M \text{ is a TM and } M \text{ accepts } w\}$

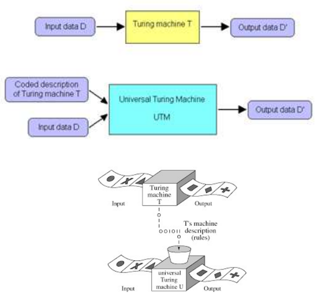
- L^A_{TM} is Turing recognizable: consider TM U which on input $< M, w >$ simulates M on w and accepts if M accepts and rejects if M rejects.

Theorem

L^A_{TM} is undecidable.

14

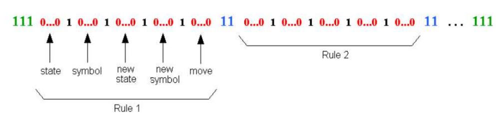
Universal Turing Machine



- A single Turing machine, properly programmed, can simulate any other Turing machine. Such a machine is called a **Universal Turing Machine (UTM).**
- The UTM accepts a coded description of a Turing machine and simulates the behavior of the machine on the input data.
- The coded description acts as a program that the UTM executes.
- The UTM's own internal program is fixed.
- The existence of the UTM is what makes computers fundamentally different from other machines such as telephones, CD players, VCRs, refrigerators, toaster-ovens, or cars.
- Computers are the only machines that can simulate any other machine to an arbitrary degree of accuracy.

15

Encoding a Turing Machine



Example: Looper TM

States: 1, 2 --> 0, 00
Symbols: **0**, **1**, **b** --> 0, 00, 000
Moves: L, R --> 0, 00

Rule 1: 1 **0** 1 **0** R --> 0101010100
Rule 2: 1 **1** 1 **1** L --> 01001010010
Rule 3: 1 **b** 2 **b** R --> 010001001000100

1110101010100110100101001011010001001000100111

16

Proof of Undecidability

Suppose $L_{TM}^{A} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ was decidable.

1. Let H be the deciding TM: on input $\langle M, w\rangle$,

$$H(\langle M, w\rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ accepts } w \\ \textit{reject} & \text{if } M \text{ does not accept } w \end{cases}$$

2. Construct TM D which on input $\langle M\rangle$, runs H on input $\langle M, \langle M\rangle\rangle$ and outputs opposite of H.

$$D(\langle M\rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ does not accept } \langle M\rangle \\ \textit{reject} & \text{if } M \text{ accepts } \langle M\rangle \end{cases}$$

3. Finally, run D with its own description $\langle D\rangle$ as input!

$$D(\langle D\rangle) = \begin{cases} \textit{accept} & \text{if } D \text{ does not accept } \langle D\rangle \\ \textit{reject} & \text{if } D \text{ accepts } \langle D\rangle \end{cases}$$

Study this topic from the other presentation on NP-Completeness!!!

17

End of Presentation

18