Turing's Thesis

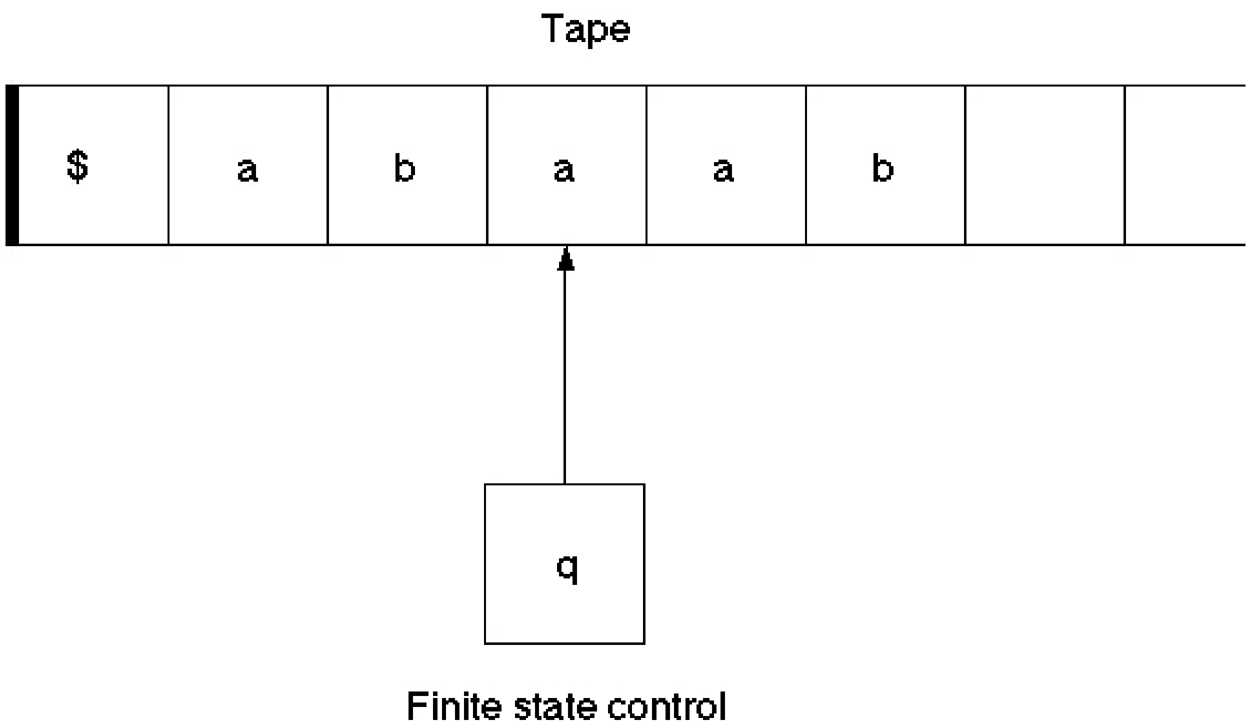
Any process which could naturally be called an effective procedure can be realized by a Turing machine.

Key points of Turing's argument:

1. One dimensional paper (divided into squares) is not a restriction.
2. Behaviour at a given moment determined by observed symbols and 'state of mind'.
3. Have a bound on number of symbols (squares) computer can observe at one moment.
4. Finite alphabet.
5. Finitely many states.
6. 'Atomic' operations: change one (observed) symbol, change observed square(s), change state.

Turing machines

Pictorial view:



Formal definition:

$$M = (Q, \Gamma, \Sigma, \not{b}, q_I, q_F, \delta),$$

where

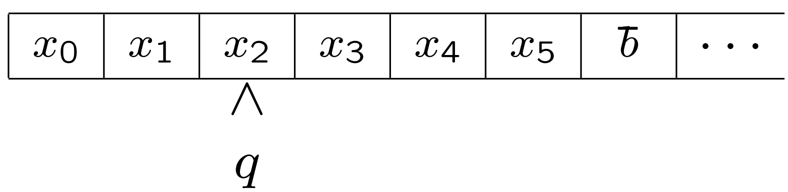
$$\begin{array}{rl} Q & \text{is a finite set of } \textit{states}, \\ \Gamma & \text{is a finite } \textit{tape alphabet}, \\ \Sigma \subset \Gamma & \text{is the } \textit{input alphabet}, \\ \not{b} \in \Gamma - \Sigma & \text{is the } \textit{blank symbol}, \\ q_I \in Q & \text{is the } \textit{initial state}, \\ q_F \in Q & \text{is the } \textit{final state}, \end{array}$$

and

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

is the *transition function* (a partial function).

Configurations:



Represented as

$$x_0 x_1 q x_2 x_3 x_4$$

Give all the relevant information about the current disposition of the machine.

Computation

Start: $q_I x_1 x_2 \cdots x_n$

Moves:

$$x_1 x_2 \cdots x_{i-1} q x_i x_{i+1} \cdots x_n \vdash x_1 x_2 \cdots x_{i-1} y q' x_{i+1} \cdots x_n$$

if $\delta(q, x_i) = (q', y, R)$.

$$x_1 x_2 \cdots x_{i-1} q x_i x_{i+1} \cdots x_n \vdash x_1 x_2 \cdots x_{i-2} q' x_{i-1} y x_{i+1} \cdots x_n,$$

if $\delta(q, x_i) = (q', y, L)$ and $i > 1$.

In all other cases the machine halts.

Input accepted: if we reach q_F.

Input rejected: if we halt in any other state *or* fall off the left end *or* never halt.

Note the asymmetry!

Derivation: γ_0, γ_1 configurations.

- $\gamma_0 \vdash \gamma_1$ means γ_1 follows from γ_0 in one move,

- $\gamma_0 \vdash^* \gamma_1$ means γ_1 follows from γ_0 in zero or more moves.

TM's as acceptors: The *language* accepted by M is

$$L(M) = \{x \in \Sigma^* \mid q_I x \vdash^* \alpha q_F \beta, \text{ where } \alpha, \beta \in \Gamma^*\}.$$

TM's as transducers: Can view M as computing a (partial) function $\Sigma^* \to \Sigma^*$; output is content of tape up to (but not including) first symbol not in Σ if machine halts. Otherwise function is undefined for the given input.

Example:

Input: A string of 0's and 1's.

Output: Accept if and only if the string is of the form 0^n1^n for some $n \geq 0$.

Examples:

1. empty string is accepted.
2. 0 is rejected.
3. 0011 is accepted.

High level description of algorithm:

while there are unmarked 0's
   mark the next 0
   find a matching 1
   if found then mark
end
if whole string marked then 'halt and accept'
else halt and reject'

Snapshots:

$$
\begin{aligned}
00\cdots011\cdots1\not{b} &\rightarrow A0\cdots011\cdots1\not{b} \\
&\rightarrow A0\cdots0B1\cdots1\not{b} \\
&\vdots \\
&\rightarrow AA\cdots ABB\cdots B\not{b}
\end{aligned}
$$

Turing machine:

$$\begin{aligned}\Gamma &= \{\, 0, 1, A, B, \not{b} \,\},\\ \Sigma &= \{\, 0, 1 \,\},\\ q_I &= \text{start},\\ q_F &= \text{accept}.\end{aligned}$$

Instructions, i.e., δ; use convention that

$$(q, x, q', y, D)$$

represents

$$\delta(q, x) = (q', y, D).$$

$(\text{start}, \not b, \text{accept}, \not b, R)$, ▷ move right important
$(\text{start}, 0, \text{findone}, A, R)$, ▷ begin loop

$(\text{findone}, 0, \text{findone}, 0, R)$,
$(\text{findone}, 1, \text{found}, B, L)$,
$(\text{findone}, B, \text{findone}, B, R)$,

$(\text{found}, B, \text{found}, B, L)$, ▷ find first unmarked 0
$(\text{found}, 0, \text{found}, 0, L)$,
$(\text{found}, A, \text{start}, A, R)$, ▷ start loop all over

$(\text{start}, B, \text{end}, B, R)$, ▷ loop now finished

$(\text{end}, B, \text{end}, B, R)$,
$(\text{end}, \not b, \text{accept}, \not b, L)$, ▷ all marked so accept