Turing Machines

Note Title 3/4/2015

A task a FSM cannot do:
determine if a binary string has more 1's than 0's.

Turing Machine meant to model anything that can be computed with any computational device.
Can compute anything you can compute with a laptop.

Simplicity of computing device $\longleftrightarrow$ Ease of programming.

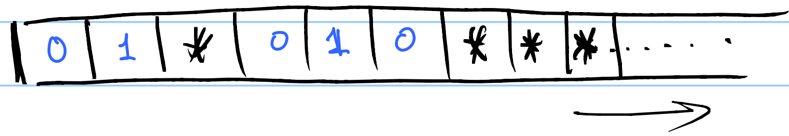
Finite tape alphabet : Γ set of symbols
must include "blank" : (*)

Finite input alphabet : Σ $\quad \Sigma \subseteq \Gamma \quad * \notin \Sigma$.

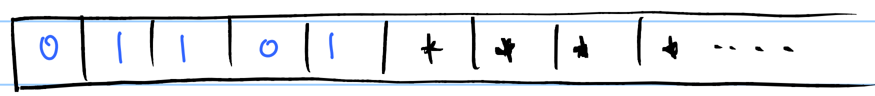
Input to the TM is $x \in \underbrace{\Sigma^*}_{\text{set of all strings over } \Sigma.}$ String of symbols from Σ.

For example, if $\Sigma = \{0, 1\}$
Σ^* is the set of all finite binary strings
$= \{ \epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots \}$

Memory = 1-dimensional tape.
infinite in one direction.
discrete cells – can hold a single character from a tape alphabet.



Starting configuration of the tape:

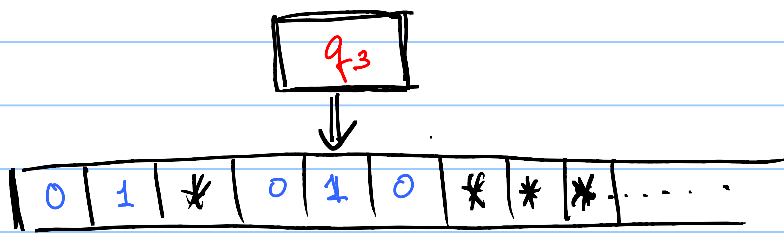


input

infinite sequence of "blank".

TM has a finite set of states: $\{q_0, q_1, \ldots, q_k\} = S$.

Current configuration has a
* current state
* location of a pointer called a "head".



Each step is dictated by a transition function:

$$\delta(q_3, 1) = (q_4, 0, L/R)$$

q_3: current state

1: tape symbol at head.

q_4: new state

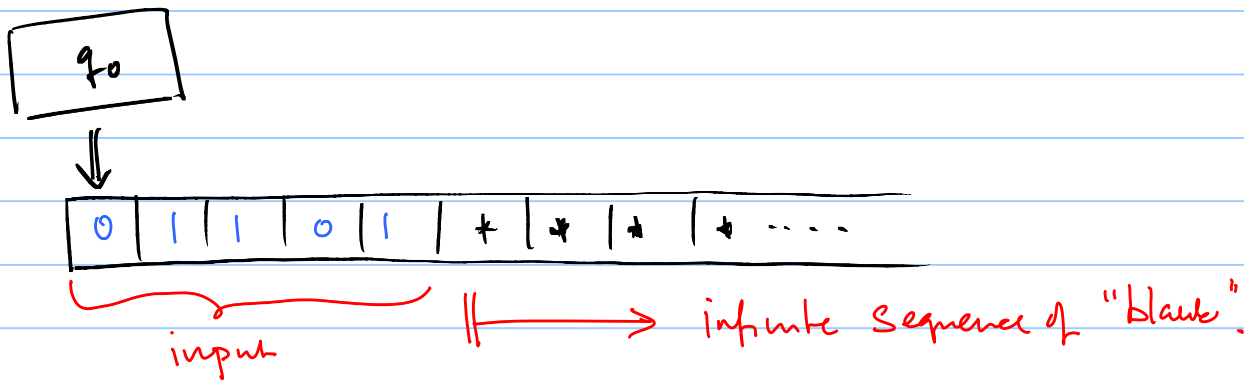
0: new symbol written

L/R: head moves left or right by 1 position.

(animation).

Three special states in S:

q_0 : start state.
q_{acc} : accept state
q_{rej} : reject state.



TM keeps executing steps until it reaches q_{acc} or q_{rej}

↳ no "next step" from these states.

$$\delta(q_3, 1) = (q_4, 0, L/R)$$

current state

tape symbol at head.

new state

new symbol written

↳ head moves left or right by 1 position.

$$\delta : S - \{q_{acc}, q_{rej}\} \times \Gamma \longrightarrow S \times \Gamma \times \{L, R\}$$

If final state is q_{acc}: accept input string
If final state is q_{rej}: reject output string.

TM could "recognize" if there are more 1's than 0's in an input string by accepting strings that do have more 1's than 0's and rejecting ones that don't.

Decision vs. Search Problems

Decision problems: output Yes / No

Is $5 \times 3 = 16$?

Is there a path from a to b in G?

Search / Computation

Compute 5×3

Find a path from a to b in G.

Input must be a string
↳ just need to have a fixed input format.

Example 1: $\Sigma = \{a, b\}$.
$\Gamma = \{a, b, *\}$

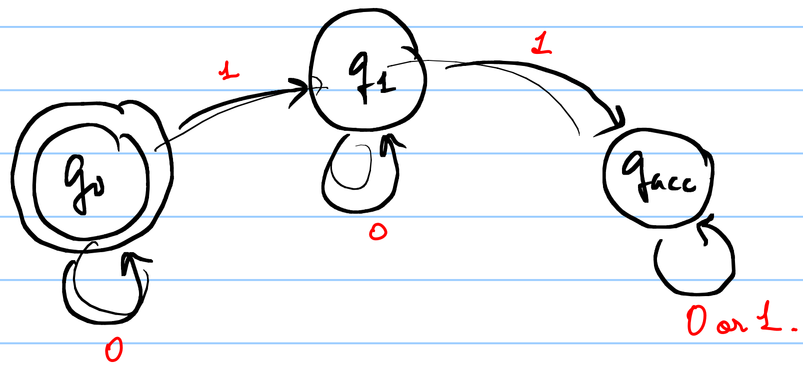
Accept $x \in \Sigma^*$ iff x has at least two b's.

$Q = \{q_0, q_1, q_{acc}, q_{rej}\}$.
└ # b's seen so far.

δ	a	b	*
q_0	(q_0, a, R)	(q_1, b, R)	$(q_{rej}, *, L)$
q_1	(q_1, a, R)	(q_{acc}, b, R)	$(q_{rej}, *, L)$.

$\delta(q_1, b) = (q_{acc}, b, R)$.

(This task could have been computed by an FSM).



More complex TM $\quad \Sigma = \{a\}$
$\Gamma = \{a, x, *\}$

Accepts $x \in \Sigma^*$
iff the # characters in x is a power of 2.

if n is a power of 2, can keep dividing n by 2 until you reach 1 with no remainder.

$8 \to 4 \to 2 \to 1$

$\cancel{12} \to 6 \to 3$

Head shuttles back and forth between the two ends of the string

→ change every other 'a' to 'x'
→ reject if there is an odd # of a's.

q_{first}, q_{even}, q_{odd}.

q_{first} → seen one a

q_{even} → seen an even # of a's

q_{odd} → seen an odd # of a's (not one).