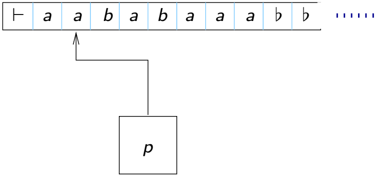
Introduction to Turing Machines

Meenakshi D'Souza

International Institute of Information Technology
Bangalore.

Term II 2022-23

How a Turing machine works



- Finite control
- Tape infinite to the right
- Each step: In current state p, read current symbol under the tape head, say a: Change state to q, replace current symbol by b, and move head left or right.

$$(p, a) \rightarrow (q, b, L/R).$$

How a Turing machine works

- Special designated accept state t and reject state r. These states are assumed to be "sink" states.
- TM accepts its input by entering state t.
- TM rejects its input by entering state r.
- TM never falls off the left end of the tape (i.e it always moves right on seeing '$\vdash$').

Example TM for $a^n b^n c^n$

Design a TM that accepts $\{a^n b^n c^n | n \geq 1\}$.

TM for adding numbers in unary

Design a TM that accepts $\{1^m\#1^n\#1^{m+n} \mid m, n \geq 0\}$.

Turning machines more formally

A Turing machine is a structure of the form

$$M = (Q, \Sigma, \Gamma, s, \delta, \vdash, \flat, t, r)$$

where

- Q is a finite set of states,
- Σ is the input alphabet,
- Γ is the tape alphabet which contains Σ,
- $s \in Q$ is the start state,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the (deterministic) transition relation,
- $\vdash \in \Gamma$ is the left-end marker.
- $\flat \in \Gamma$ is the blank tape symbol.
- $t \in Q$ is the accept state.
- $r \in Q$ is the reject state.

Configurations, runs, etc. of a Turing machine

- A configuration of M is of the form $(p, y\flat^\omega, n) \in Q \times \Gamma^\omega \times \mathbb{N}$, which says "$M$ is in state p, with "non-blank" tape contents y, and read head positioned at the n-th cell of the tape".
- Initial configuration of M on input w is $(s, \vdash w\flat^\omega, 0)$.
- 1-step transition of M: If $(p, a) \to (q, b, L)$ is a transition in δ, and $z(n) = a$: then

$$(p, z, n) \stackrel{1}{\Rightarrow} (q, s_b^n(z), n-1).$$

- Similarly, if $(p, a) \to (q, b, R)$ is a transition in δ, and $z(n) = a$: then

$$(p, z, n) \stackrel{1}{\Rightarrow} (q, s_b^n(z), n+1).$$

- M accepts w if $(s, \vdash w\flat^\omega, 0) \stackrel{*}{\Rightarrow} (t, z, i)$, for some z and i.
- M rejects w if $(s, \vdash w\flat^\omega, 0) \stackrel{*}{\Rightarrow} (r, z, i)$, for some z and i.

Language accepted by a Turing machine

- The Turing machine M is said to halt on an input if it eventually gets into state t or r on the input.
- Note that M may not get into either state t or r on a particular input w. In that case we say M loops on w.
- A machine that halts on all inputs is called a total Turing machine.
- The language accepted by M is denoted $L(M)$ and is the set of strings accepted by M.
- A language $L \subseteq \Sigma^*$ is called recursively enumerable if it is accepted by some Turing machine M.
- A language $L \subseteq \Sigma^*$ is called recursive if it is accepted by some Turing machine M which halts on all inputs.

Recursive sets: Closure under complement

- Recursive sets are closed under complement.
- Proof: Suppose A is recursive. Then, there exists a total TM M such that $L(M) = A$. Define M' to be the same TM as M, but, with accept and reject states swapped. It is clear that $L(M') = A^C$, where A^C denotes the complement of A.

Recursive and r.e. sets

- Every recursive set is r.e. but not necessarily vice versa.
- If both A and A^C are r.e. then A is recursive.
- Proof: Suppose both A and A^C are r.e. Let M and M' be TMs such that $L(M) = A$ and $L(M') = A^C$. Build a new machine N that simulates M and M' on two of its tracks. N alternately performs a step of M and a step of M', shuttling back and forth between the two simulated tape head positions of M and M', and updating the tape. Transiton details of M and M' are stored in N's finite control.
  If M accepts, then so does N. If M' rejects, then N accepts. Exactly one of these must occur, depending on whether $w \in A$ or $w \in A^C$. Thus N halts on all inputs and $L(N) = A$.

Decidability and Semi-decidability

- A property P of strings is said to be decidable if the set of all strings having property P is a recursive set.
- A property P of strings is said to be semi-decidable if the set of all strings having property P is an r.e. set.
- The notion of decidablility (semi-decidability) is equivalent to recursive (r.e.).

$$
\begin{gathered}
P \text{ is decidable} \Leftrightarrow \{x \mid P(x)\} \text{ is recursive.} \\
A \text{ is recursive} \Leftrightarrow x \in A \text{ is decidable.} \\
P \text{ is semi-decidable} \Leftrightarrow \{x \mid P(x)\} \text{ is r.e.} \\
A \text{ is r.e} \Leftrightarrow x \in A \text{ is semidecidable.}
\end{gathered}
$$