

# Universal Models of Computation

## Universal Models of Computation

- Models of computation that have power equivalent to that of an idealized general-purpose computer or, models of computation characterize problem-solving by humans and machines.
  - how is the input (output) represented?
  - how does the computational model compute?
  - what is the cost (in time and space) ?
- Use models to determine what can and cannot be computed.

## Issues of Computability

- “What problems can be solved on a given model of computation?”
- *Halting Problem* — a classical unsolvable problem:
  - “Does there exist an algorithm which, given two bit string, the first is a program and the second is data, determine where or not the program run on the data will stop?”
- Another one: “Does the program return the correct answer?”

## A Contradiction Argument for Halting Problem

- Assume an algorithm for the Halting Problem exists. Let it be  $P$ .

boolean function  $P$  (string prog, data)  
{ ... }

$P$  returns  $T$  (“prog” stops on “data”) or  $F$  (“prog” doesn’t stop)

```
procedure  $Q$  (string  $x$ )  
{ if (! $P(x, x)$ ) exit(0);  
  while (1) { }; }
```

- $Q(Q)$  ?  
 $Q(Q)$  stops iff  $P(Q, Q)$  returns  $F$ , which happens iff  $Q$  doesn’t stop on  $Q$ .  
 $Q(Q)$  doesn’t stop iff  $P(Q, Q)$  returns  $T$ , which happens iff  $Q$  stops on  $Q$ .  
Contradiction!

## Turing Machines

- Proposed by Alan Turing in 1936.
- Mimic the problem-solving mechanism of a scientist
  - unbounded supply of paper, pencils, erasers and thinks.
  - write/lookup/modify notes
  - a finite number of mental states: finite volume of brain and discrete thought processes.
  - make decision based on notes and the scientist's current mental state
- TM: has an unlimited and unrestricted memory (an infinite tape).
  - read/write tape
  - move left/right
  - rejecting/accepting states take immediate effect.
- Equivalent in power with a real computer.

## An Example

- A Turing machine recognizes  $L = \{w\#w \mid w \in \{0, 1\}^*\}$ .
  - multiple passes over the input string.
  - match symbols of both side of  $\#$  one by one.

## Formal Definition of a TM

- A TM is a 7-tuple  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$  where
  - $Q$ : finite set of states
  - $\Sigma$ : finite input alphabet (not containing the blank symbol)
  - $\Gamma$ : finite tape alphabet ( $\Sigma \subseteq \Gamma$ )
  - $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$   
e.g.:  $\delta(q, a) = (r, b, L)$
  - $q_0 \in Q$ : start state
  - $q_{accept} \in Q$ : the accept state
  - $q_{reject} \in Q$ : the reject state

## Operations of TM

- Copy input to tape.
- Start on the leftmost symbol on the tape.
- Proceed according to the transition rules.  
(Never move off the left end of the tape.)
- Continue until it  
enters the accept state,  
or the reject state,  
or goes on forever.

## Configurations of TM

- Configuration: a snapshot
    - current state
    - current tape contents
    - current head location
- e.g.  $uqv$ ,  
10110 $q_1$ 0110
- Start configuration:  $q_0w$
  - Halting configurations
    - accepting configuration: accept state
    - rejecting configuration: reject state

## Language of TM

- Yield: configuration  $C_1$  yields  $C_2$  if the TM can go from  $C_1$  to  $C_2$  in one step.  
e.g. If  $\delta(q, a) = (r, b, L)$ , then  $ubqav$  yields  $urbarv$
- A TM  $M$  accepts a string  $w$ :  
 $M$  goes from the start configuration to an accepting configuration on input  $w$ .
- The language of  $M$ :  
 $L(M) = \{ w \mid w \in \Sigma^* \text{ and } M \text{ accepts } w \}$