# The Church-Turing Thesis

*Foundations of Computing Science*

**Pallab Dasgupta**
**Professor,**
**Dept. of Computer Sc & Engg**

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

# Turing Machines

A Turing Machine is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where

- $Q, \Sigma, \Gamma$ are all *finite* sets
- $Q$ is the set of *states*
- $\Sigma$ is the *input alphabet* not containing the *blank symbol* $\sqcup$
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the *transition function*
- $q_0 \in Q$ is the *start state*
- $q_{accept} \in Q$ is the *accept state*
- $q_{reject} \in Q$ is the *reject state*, where $q_{reject} \neq q_{accept}$

Differences Between Finite Automata and Turing Machines

- A Turing machine can both write on the tape and read from it.
- The read-write head can both move to the left and to the right.
- The tape is infinite.
- The special states for rejecting and accepting take effect immediately.

# Language for Turing Machines

A Turing Machine *M* *accepts* input *w* if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where

- $C_1$ is the *start configuration* of *M* on input *w*
- Each $C_i$ *yields* $C_{i+1}$
- $C_k$ is an *accepting configuration*

The collection of strings that *M* accepts is the *language of M*, or the *language recognized by M*, denoted by *L(M)*

- A language is Turing-recognizable (recursively enumerable language) iff some Turing machine recognizes it.
- A language is Turing-decidable (recursive language) or decidable iff some Turing machine decides it.
- Every Turing-decidable language is also Turing-recognizable.

# Example of Turing Machine

Turing Machine *M* that decides $A = \{0^{2^n} \mid n \geq 0\}$

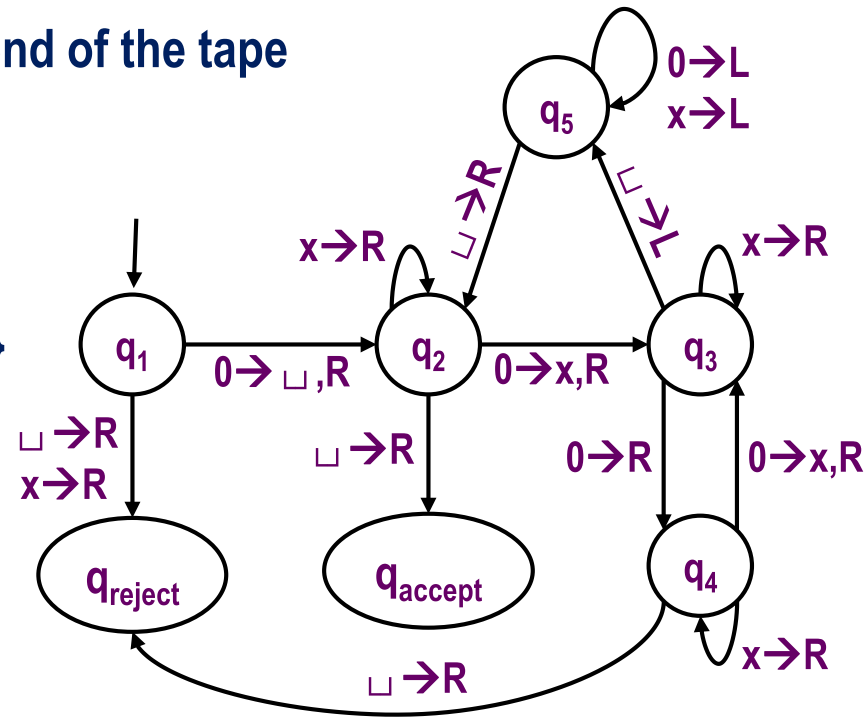
*M* on input string *w*,

1. Sweep left to right across the tape, crossing off every alternate 0
2. If in Stage 1, the tape contained a single 0, *accept*
3. If in Stage 1, the tape contained more than a single 0, and the number of 0s was odd, *reject*
4. Return the head to the left-hand end of the tape
5. Go to Stage 1

$M = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$, where

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$
- $q_1 \leftarrow$ start state
- $q_{accept} \leftarrow$ accept state
- $q_{reject} \leftarrow$ reject state
- $\Sigma = \{0\}$ and $\Gamma = \{0, x, \sqcup\}$

$\boxed{\delta}$



# Variants of Turing Machines

## Multi-tape Turing Machines

- *Transition function*, $\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$ (*k* is the number of tapes)

- The expression $\delta(q_i, a_1, a_2, \ldots, a_k) = (q_j, b_1, b_2, \ldots, b_k, L, S, \ldots, L)$ means that,
  - the machine is in state $q_i$ and heads *1* through *k* reading $a_1$ through $a_k$
  - the machine goes to state $q_j$, writing symbols $b_1$ through $b_k$
  - the machine directs each head to move *left(L)*, *right(R)* or *stay put(S)*

- Theorems:
  - Every multi-tape Turing machine has an equivalent single-tape Turing machine
  - A language is Turing-recognizable if and only if some multi-tape Turing machine recognizes it

# Variants of Turing Machines (contd…)

## Nondeterministic Turing Machines

- *Transition function*, $\delta: Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$
- Theorems:
  - Every nondeterministic Turing machine has an equivalent deterministic Turing machine
  - A language is Turing-recognizable if and only if some nondeterministic Turing machine recognizes it
  - A language is decidable if and only if some nondeterministic Turing machine decides it

## Enumerators

- Starts with a blank input tape
- Language can be infinite list of strings (may be repetitive and generated by the enumerator without halting)
- Theorem: A language is Turing-recognizable if and only if some enumerator enumerates it

# The Definition of Algorithm

- **Hilbert's 10th Problem:**
  - Devise an algorithm that tests whether a polynomial has an integral root.
  - D = {p | p is a polynomial with an integral root} – whether *D* is *decidable*
  - The answer is *negative*; in contrast *D* is *Turing-recognizable*

- **To Define Algorithms**
  - Church proposed notational system called λ-calculus
  - Turing proposed "machine"

  equivalent

- **The Church-Turing Thesis**

| | | |
|---|---|---|
| Intuitive notion of algorithms | equals | Turing machine algorithms |