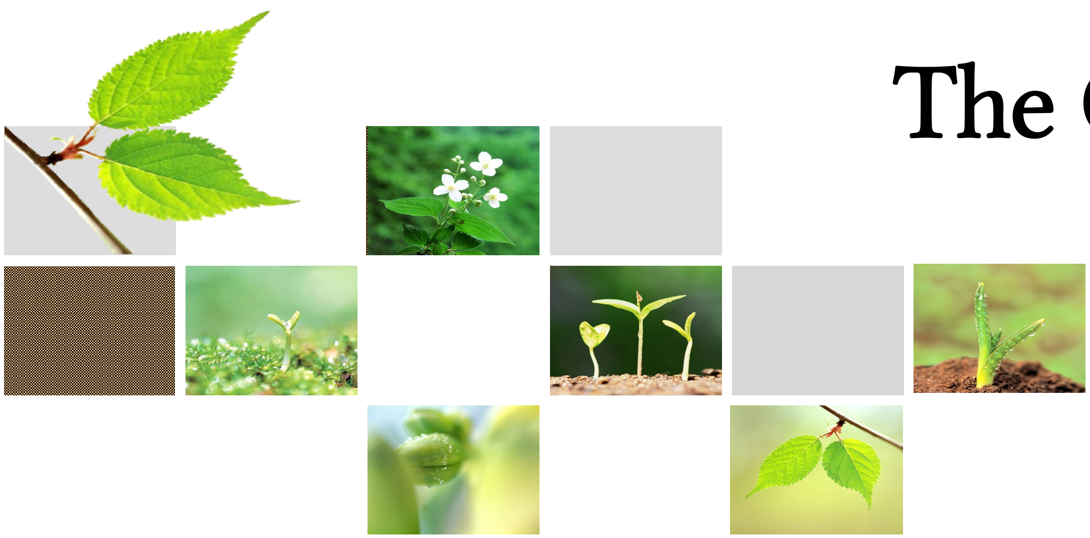
بسم الله الرحمن الرحیم



The Church-Turing Thesis

Ali Shakiba – Vali-e-Asr University of Rafsanjan – ali.shakiba@vru.ac.ir

We are going to discuss …

1. Turing Machines
2. Variants of T.M.s
3. What’s an “Algorithm”?

Turing Machine

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

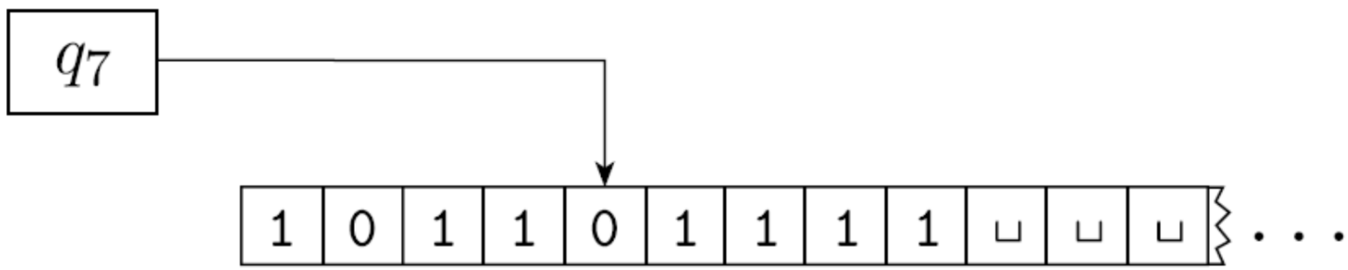
1. Q is the set of states
2. Σ is the **input alphabet** such that **blank symbol** $\sqcup \notin \Sigma$
3. Γ is the **tape alphabet** such that $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
5. $q_0 \in Q$ is the **start state**
6. $q_{accept}, q_{reject} \in Q$ are accept and reject states where $q_{accept} \neq q_{reject}$

Turing Machine

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

1. Q is the set of states
2. Σ is the **input alphabet** such that **blank symbol** $\sqcup \notin \Sigma$
3. Γ is the **tape alphabet** such that $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. $\delta: Q' \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ where $Q' = Q \setminus \{q_{accept}, q_{reject}\}$
5. $q_0 \in Q$ is the **start state**
6. $q_{accept}, q_{reject} \in Q$ are accept and reject states where $q_{accept} \neq q_{reject}$

Configuration



$$10111\,q_7\,01111$$

Yields


- Assume
  - $a, b, c \in \Gamma, u, v \in \Gamma^*, q_i, q_j \in Q$
  - uaq_ibv is a configuration

$u a \boldsymbol{q_i} b v$ yields $u \boldsymbol{q_j} a c v$

if

$\delta(q_i, b) = (q_j, c, L)$

$u a \boldsymbol{q_i} b v$ yields $u a c \boldsymbol{q_j} v$

if

$\delta(q_i, b) = (q_j, c, R)$

Some designated configurations

- Start configuration
  - $q_0 w$
- Accepting configuration
  - The configuration is in state q_{accept}
- Rejecting configuration
  - The configuration is in state q_{reject}

Halting Configurations

TM M Accepts w

- TM M accepts input w if a sequence of configurations $C_1, \ldots, C_k$ exists where
  1. C_1 is the start configuration of M on input w
  2. each C_i yields C_{i+1}, and
  3. C_k is an accepting configuration

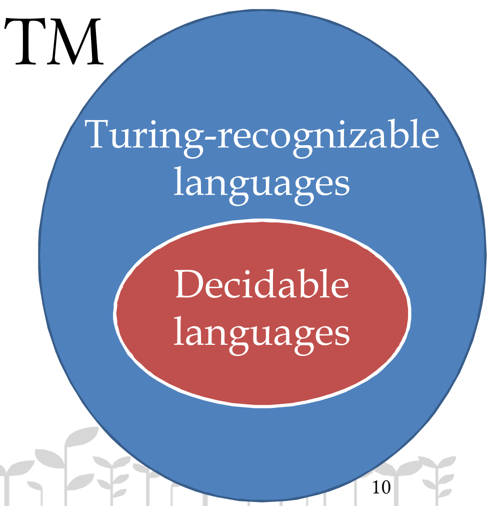
Turing–recognizable and –decidable languages

A language L is Turing-recognizable (recursively-enumerable) if some TM

1. accept strings in L, and
2. rejects strings not in L by entering q_{reject} or looping

A language L is Turing-decidable (recursive) if some TM

1. accept strings in L, and
2. rejects strings not in L by entering q_{reject}



Some Examples

Describe a TM M_2 to decide $A = \{0^{2^n} \mid n \geq 0\}$

$M_2 =$ "On input string w:

1. Sweep left to right across the tape, crossing off every other 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."

Some Examples (cont'd)

Describe a TM M_3 to decide $C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

$M_3 =$ "On input string w:

1. Scan the input from left to right to determine whether it is a member of $a^+b^+c^+$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and the c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*.
4. Restore the crossed off b's and repeat stage 3 if there is another a to cross off. If all a's have been crossed off, determine whether all c's also have been crossed off. If yes, *accept*; otherwise, *reject*."

Marking Tape Symbols Technique

Some Examples (cont'd)

Describe a TM M_4 to decide

$$E = \{\#x_1\#x_2\# \cdots \#x_\ell \mid \text{each } x_i \in \{0,1\}^* \text{and } x_i \neq x_j \text{ for each } i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.
5. Go to stage 3."