Computational Theory

Turing Machines

Curtis Larsen

Utah Tech University—Computing

Fall 2024

Adapted from notes by Russ Ross

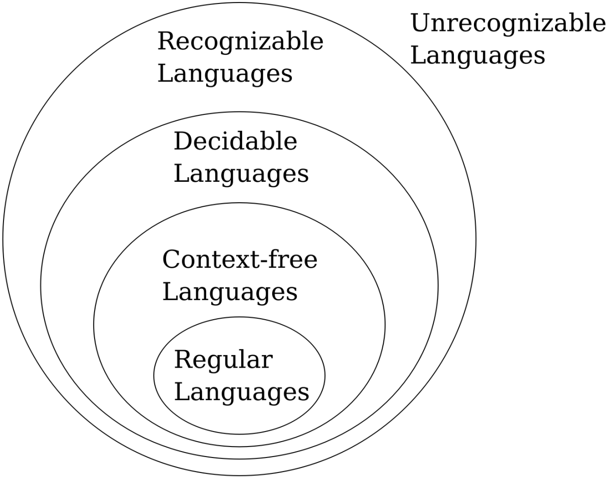
Turing Machines

Reading: Sipser §3.1.

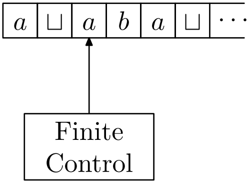
Status Update

- Regular languages: DFA, NFA, RE, PL for RL
- Context-free languages: CFG, PDA, PL for CFL
- Turing Machines:
  - Decidable languages
  - Recognizable languages
  - Unrecognizable languages

Status Update



The Basic Turing Machine



- Head can both read and write, and move in both directions.
- Tape has a beginning on the left, and unbounded length.
- ⊔ is the blank symbol. All but a finite number of tape squares are blank.
- Accept and reject states take effect immediately, not waiting for end of input.

Formal Definition of a TM

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

- Q is a finite set of states
- Σ is the finite **input alphabet**; $\sqcup \notin \Sigma$
- Γ is the finite **tape alphabet**; $\sqcup \in \Gamma$, $\Sigma \subset \Gamma$
- δ: $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- $q_0 \in Q$ is the **start state**
- $q_{\text{accept}} \in Q$ is the **accept state**
- $q_{\text{reject}} \in Q$ is the **reject state**; $q_{\text{reject}} \neq q_{\text{accept}}$

The transition function

$$Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

- L and R are “move left” and “move right”
- $\delta(q, b) = (r, c, R)$
  - Rewrite b as c in current cell
  - Switch from state q to state r
  - And move right
- $\delta(q, b) = (r, c, L)$
  - Same as R, but move left
  - *Unless* at left end of tape, in which case stay put

Computation of TMs

- A **configuration** is uqv, where $q \in Q$, $u, v \in \Gamma^*$.
  - Tape contents $= uv$ followed by all blanks
  - State $= q$
  - Head on first symbol of v.
  - Don't explicitly write the infinite number of $\sqcup$ at the end of v.
- Start configuration $= q_0 w$, where w is input.
- One step of computation: (configuration C_i yields C_{i+1})
  - Configuration $= uaqbv$; $u, v \in \Gamma^*$; $a, b \in \Gamma$; $q \in Q$.
  - $uaqbv \to uacrv$, if $\delta(q, b) = (r, c, R)$; $b, c \in \Gamma$; $q, r \in Q$.
  - $uaqbv \to uracv$, if $\delta(q, b) = (r, c, L)$; $b, c \in \Gamma$; $q, r \in Q$.
  - $qbv \to rcv$, if $\delta(q, b) = (r, c, L)$; $b, c \in \Gamma$; $q, r \in Q$.
- If $r \in \{q_{\text{accept}}, q_{\text{reject}}\}$, computation halts.

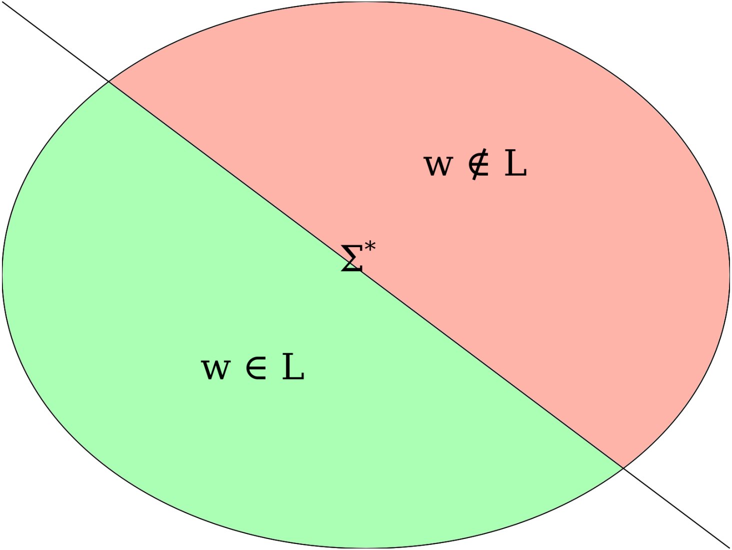
TM Results

- M **accepts** w if there is a sequence of configurations $C_1, \ldots, C_k$ such that
  1. $C_1 = q_0 w$.
  2. C_i yields C_{i+1} for each i.
  3. C_k is an accepting configuration (i.e. state of M is q_{accept}).
- M **rejects** w if there is a sequence of configurations $C_1, \ldots, C_k$ such that
  1. $C_1 = q_0 w$.
  2. C_i yields C_{i+1} for each i.
  3. C_k is a rejecting configuration (i.e. state of M is q_{reject}).
- M **halts** on w if it **accepts** or **rejects** w.
- M **loops** on w if it does not **halt** on w.

TMs and Language Membership

- $L(M) = \{w | M \text{ accepts } w\}$.
- L is **Turing-recognizable** if $L = L(M)$ for some TM M, and:
  - $w \in L \Rightarrow M$ halts on w in state q_{accept}.
  - $w \notin L \Rightarrow M$ halts on w in state q_{reject} OR M never halts (it “loops”).
- L is **(Turing-)?decidable** if $L = L(M)$ for some TM M, and:
  - $w \in L \Rightarrow M$ halts on w in state q_{accept}.
  - $w \notin L \Rightarrow M$ halts on w in state q_{reject}.

$w \in L$ or $w \notin L$



Example Language

- $B = \{w\#w | w \in \{0,1\}^*\}$
- B is not context-free. (Can be shown with CFL PL)
- B is decidable. (Can be shown with TM)

Formal Descriptions

Formal description of M_B, where $L(M_B) = B$.

- $Q = \{q_0, ..., q_{\text{accept}}, q_{\text{reject}}\}$
- $\Sigma = \{0, 1, \#\}$
- $\Gamma = \{0, 1, \#, x, \sqcup\}$
- δ: ...

OR state diagram.

Implementation-level Descriptions

Let $M_B =$ "On input string w:

1. Until $\#$ is read.
2. Remember the symbol read, write x.
3. Move right until $\#$ or $\sqcup$ seen.
4. If $\sqcup$, *reject*.
5. Move right while x seen.
6. If symbol read is $\sqcup$ or not remembered symbol, *reject*.
7. Write x.
8. Move left until $\#$.
9. Move left until x.
10. Move right.
11. Move right until something other than x is read.
12. If symbol read is $\sqcup$, *accept*. Otherwise, *reject*."

High-level Descriptions

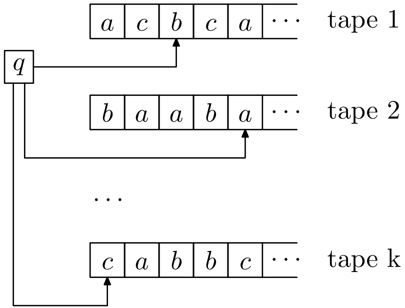
Let $M_B =$ "On input string w:

1. If there is no $\#$, *reject*.
2. For each symbol left of the $\#$, match against same position right of the $\#$. If there is a mismatch, *reject*.
3. If there are extra non-blank symbols right of the $\#$, *reject*.
4. *accept*."

Multitape Machines

For a k tape machine:

δ: $Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$



Multitape Machines

Theorem 3.13

Every multitape Turing machine has an equivalent single-tape Turing machine.

Proof?

Nondeterministic Machines

$\delta\colon Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$

Nondeterministic Machines

Theorem 3.16

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Proof?

The Church-Turing Thesis

Figure 3.22

Our ***intuitive notion of algorithms*** is equal to ***Turing machine algorithms***.

Sample problem

Let $A = \{\langle G \rangle | G$ is a connected undirected graph $\}$.

Is A decidable?

Sample problem

Let $A = \{\langle G \rangle | G$ is a connected undirected graph $\}$.

Is A decidable?

Let $M =$"On input $\langle G \rangle$, the encoding of a graph:

1. Select the first node of G and mark it.
2. Repeat the following state until no new nodes are marked:
3. For each node in G, mark it if it is attached by an edge to a node that is already marked.
4. Scan all nodes of G to determine whether they are all marked. If the are *accept*; otherwise *reject*."

- **DEFINITION 3.3** Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

- **DEFINITION 3.3** Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **DEFINITION 3.5** Turing-recognizable.

- **DEFINITION 3.3** Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **DEFINITION 3.5** Turing-recognizable.
- **DEFINITION 3.6** Turing-decidable (decidable).

- **DEFINITION 3.3** Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **DEFINITION 3.5** Turing-recognizable.
- **DEFINITION 3.6** Turing-decidable (decidable).
- **THEOREM 3.13** Every multi-tape Turing machine has an equivalent single-tape Turing machine.

- **DEFINITION 3.3** Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **DEFINITION 3.5** Turing-recognizable.
- **DEFINITION 3.6** Turing-decidable (decidable).
- **THEOREM 3.13** Every multi-tape Turing machine has an equivalent single-tape Turing machine.
- **COROLLARY 3.15** A language is Turing-recognizable if and only if some multi-tape Turing machine recognizes it.

- **DEFINITION 3.3** Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **DEFINITION 3.5** Turing-recognizable.
- **DEFINITION 3.6** Turing-decidable (decidable).
- **THEOREM 3.13** Every multi-tape Turing machine has an equivalent single-tape Turing machine.
- **COROLLARY 3.15** A language is Turing-recognizable if and only if some multi-tape Turing machine recognizes it.
- **COROLLARY 3.XX** A language is decidable if and only if some multi-tape Turing machine decides it.

- **DEFINITION 3.3** Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **DEFINITION 3.5** Turing-recognizable.
- **DEFINITION 3.6** Turing-decidable (decidable).
- **THEOREM 3.13** Every multi-tape Turing machine has an equivalent single-tape Turing machine.
- **COROLLARY 3.15** A language is Turing-recognizable if and only if some multi-tape Turing machine recognizes it.
- **COROLLARY 3.XX** A language is decidable if and only if some multi-tape Turing machine decides it.
- **THEOREM 3.16** Every non-deterministic Turing machine has an equivalent deterministic Turing machine.

- **DEFINITION 3.3** Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **DEFINITION 3.5** Turing-recognizable.
- **DEFINITION 3.6** Turing-decidable (decidable).
- **THEOREM 3.13** Every multi-tape Turing machine has an equivalent single-tape Turing machine.
- **COROLLARY 3.15** A language is Turing-recognizable if and only if some multi-tape Turing machine recognizes it.
- **COROLLARY 3.XX** A language is decidable if and only if some multi-tape Turing machine decides it.
- **THEOREM 3.16** Every non-deterministic Turing machine has an equivalent deterministic Turing machine.
- **COROLLARY 3.18** A language is Turing-recognizable if and only if some non-deterministic Turing machine recognizes it.

- **DEFINITION 3.3** Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **DEFINITION 3.5** Turing-recognizable.
- **DEFINITION 3.6** Turing-decidable (decidable).
- **THEOREM 3.13** Every multi-tape Turing machine has an equivalent single-tape Turing machine.
- **COROLLARY 3.15** A language is Turing-recognizable if and only if some multi-tape Turing machine recognizes it.
- **COROLLARY 3.XX** A language is decidable if and only if some multi-tape Turing machine decides it.
- **THEOREM 3.16** Every non-deterministic Turing machine has an equivalent deterministic Turing machine.
- **COROLLARY 3.18** A language is Turing-recognizable if and only if some non-deterministic Turing machine recognizes it.
- **COROLLARY 3.19** A language is decidable if and only if some non-deterministic Turing machine decides it.

- **Definition 3.3** Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **Definition 3.5** Turing-recognizable.
- **Definition 3.6** Turing-decidable (decidable).
- **Theorem 3.13** Every multi-tape Turing machine has an equivalent single-tape Turing machine.
- **Corollary 3.15** A language is Turing-recognizable if and only if some multi-tape Turing machine recognizes it.
- **Corollary 3.xx** A language is decidable if and only if some multi-tape Turing machine decides it.
- **Theorem 3.16** Every non-deterministic Turing machine has an equivalent deterministic Turing machine.
- **Corollary 3.18** A language is Turing-recognizable if and only if some non-deterministic Turing machine recognizes it.
- **Corollary 3.19** A language is decidable if and only if some non-deterministic Turing machine decides it.
- **Figure 3.22** The Church-Turing Thesis states that a Turing machine and an algorithm are equivalent.