Outline

- Turing Machines and variants
  - multitape TMs
  - nondeterministic TMs
- Church-Turing Thesis
- decidable, RE, co-RE languages

Turing Machine

- Informal description:
  - input written on left-most squares of tape
  - rest of squares are blank
  - at each point, take a step determined by
    - current symbol being read
    - current state of finite control
  - a step consists of
    - writing new symbol
    - moving read/write head left or right
    - changing state

Example Turing Machine

language $L = \{w#w : w \in \{0,1\}^*\}$

Example TM diagram

Turing Machine diagrams

Example TM diagram
**TM formal definition**

- A TM is a 7-tuple \((Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\) where:
  - \(Q\) is a finite set called the **states**
  - \(\Sigma\) is a finite set called the **input alphabet**
  - \(\Gamma\) is a finite set called the **tape alphabet**
  - \(\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\) is a function called the **transition function**
  - \(q_0\) is an element of \(Q\) called the **start state**
  - \(q_{\text{accept}}, q_{\text{reject}}\) are the **accept** and **reject states**

**Example TM operation**

<table>
<thead>
<tr>
<th>tape</th>
<th>start</th>
<th>start</th>
<th>start</th>
<th>accept</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>#</td>
<td>0</td>
</tr>
</tbody>
</table>

**TM configurations**

- At every step in a computation, a TM is in a **configuration** determined by:
  - the contents of the tape
  - the state
  - the location of the read/write head

- next step completely determined by current configuration

- shorthand: string \(uqv\) with \(u,v \in \Gamma^*, q \in Q\)

**TM configurations**

- Formal definition of “yields”:
  - \(uqv \Rightarrow uqv\) if \(\delta(q, \sigma) = (q', c, L)\) and \(u'=u\)
  - \(uqv \Rightarrow uqv\) if \(\delta(q, \sigma) = (q', c, R)\)

- two special cases:
  - left end: \(qv \Rightarrow q\sigma c\) if \(\delta(q, \sigma) = (q, c, L)\)
  - right end: \(uqv\) same as \(uqv\)

**TM acceptance**

- start configuration: \(q_0w\) (\(w\) is input)
- accepting config.: any config. with state \(q_{\text{accept}}\)
- rejecting config.: any config. with state \(q_{\text{reject}}\)

TM \(M\) accepts input \(w\) if there exist configurations \(C_1, C_2, \ldots, C_k\)
- \(C_1\) is start configuration of \(M\) on input \(w\)
- \(C_i \Rightarrow C_{i+1}\) for \(i = 1, 2, 3, \ldots, k-1\)
- \(C_k\) is an accepting configuration
Deciding and Recognizing

• TM M:
  – L(M) is the language it recognizes
  – if M rejects every x \in L(M) it decides L
  – set of languages recognized by some TM is called Turing-recognizable or recursively enumerable (RE)
  – set of languages decided by some TM is called Turing-decidable or decidable or recursive

Classes of languages

• We know: regular \subseteq CFL (proper containment)
• CFL \subseteq decidable
  – proof?
  – decidable \subseteq RE \subseteq all languages
  – proof?

Multitape TMs

• A useful variant: k-tape TM
  A multitape Turing machine is a 7-tuple (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}) where:
  \delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k
  \delta(q_i, a_1, a_2, \ldots, a_k) = (q_j, b_1, b_2, \ldots, b_k, L, R, \ldots, L) =
  “in state q_i, reading a_1, a_2, \ldots, a_k on k tapes, move to state q_j, write b_1, b_2, \ldots, b_k on k tapes, move L, R on k tapes as specified.”
Multitape TMs

**Theorem:** every k-tape TM has an equivalent single-tape TM.

**Proof:**
- Idea: simulate k-tape TM on a 1-tape TM.

---

Nondeterministic TMs

- A important variant: *nondeterministic TM*
- informally, several possible next configurations at each step
- formally, a NTM is a 7-tuple

\[(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\]

where:
- everything is the same as a TM except the transition function:
  \[\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})\]

---

NTM acceptance

- start configuration: \[q_0 w\] (w is input)
- accepting config.: any config. with state \[q_{\text{accept}}\]
- rejecting config.: any config. with state \[q_{\text{reject}}\]

NTM M accepts input w if there exist configurations \(C_1, C_2, \ldots, C_k\)
- \(C_1\) is start configuration of M on input w
- \(C_i \Rightarrow C_{i+1}\) for \(i = 1, 2, 3, \ldots, k-1\)
- \(C_k\) is an accepting configuration

---

Theorem: every NTM has an equivalent (deterministic) TM.

Proof:
- Idea: simulate NTM with a deterministic TM
Nondeterministic TMs

Simulating NTM $M$ with a deterministic TM:
- computations of $M$ are a tree
- nodes are configs
- fanout is $b =$ maximum number of choices in transition function
- leaves are accept/reject configs.

Nondeterministic TMs

Simulating NTM $M$ with a deterministic TM:
- idea: breadth-first search of tree
- if $M$ accepts: we will encounter accepting leaf and accept
- if $M$ rejects: we will encounter all rejecting leaves, finish traversal of tree, and reject
- if $M$ does not halt on some branch: we will not halt...