Outline

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

Turing Machine diagrams

- Start state
- Transition label: (tape symbol read → tape symbol written, direction moved)
  - a → R means "read a, move right"
  - a → L means "read a, move left"
  - a → b, R means "read a, write b, move right"

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>State</th>
<th>q</th>
<th>$\delta(q,a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0</td>
<td>(start, 0, R)</td>
</tr>
<tr>
<td>start</td>
<td>1</td>
<td>(start, 1, R)</td>
</tr>
<tr>
<td>start</td>
<td>_</td>
<td>(L, _ , L)</td>
</tr>
<tr>
<td>start</td>
<td>#</td>
<td>(start, #, R)</td>
</tr>
<tr>
<td>t</td>
<td>0</td>
<td>(accept, 1, -)</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
<td>(L, 0, L)</td>
</tr>
<tr>
<td>t</td>
<td>#</td>
<td>(accept, #, R)</td>
</tr>
</tbody>
</table>

TM configurations

- At every step in a computation, a configuration determined:
  - 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

- configuration $C_1$ yields configuration $C_2$ if TM can legally* move from $C_1$ to $C_2$ in 1 step
  - notation: $C_1 \Rightarrow C_2$
  - also: “yields in 1 step” notation: $C_1 \Rightarrow^1 C_2$
- “yields in k steps” notation: $C_1 \Rightarrow^k C_2$
  if there exists configurations $D_1, D_2, \ldots, D_{k-1}$ for which $C_1 \Rightarrow D_1 \Rightarrow D_2 \Rightarrow \ldots \Rightarrow D_{k-1} \Rightarrow C_2$
- also: “yields in some # of steps” ($C_1 \Rightarrow^* C_2$)

*Convention: TM halts upon entering $q_{\text{accept}}$, $q_{\text{reject}}$

---

TM acceptance

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

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 $x \notin L(M)$ 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?

---

Formal definition of "yields":

- $u \alpha q_i \beta v \Rightarrow u \alpha q_j \alpha \beta v$ if $\delta(q_i, \beta) = (q_j, \alpha, L)$, and
- $u \alpha q_i \beta v \Rightarrow u \alpha q_j \beta v$ if $\delta(q_i, \beta) = (q_j, \alpha, R)$

- two special cases:
  - left end: $q_i \beta v \Rightarrow q_j \gamma v$ if $\delta(q_i, \beta) = (q_j, \gamma, L)$
  - right end: $u \alpha q_i \Rightarrow u \alpha q_j$ same as $u \alpha q_i \_ u, v \in \Gamma$

$a, b, c \in \Gamma$
$q_i, q_j \in Q$q
$q_i \neq q_{\text{accept}}, q_{\text{reject}}$
Multitape TMs

• A useful variant: k-tape TM

Informal description of k-tape TM:
– input written on left-most squares of tape #1
– rest of squares are blank on all tapes
– at each point, take a step determined by
  • current k symbols being read on k tapes
  • current state of finite control
– a step consists of
  • writing k new symbols on k tapes
  • moving each of k read/write heads left or right
  • changing state

Multitape TM formal definition

• A TM 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:
  \(\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k\)

\[\delta(q_i, a_1, a_2, \ldots, a_k) = (q_j, b_1, b_2, \ldots, b_k, L, R, \ldots, L) = \]
  “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.”

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

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

Multitape TMs

simulation of k-tape TM by single-tape TM:

• add new symbol \(x\) for each old \(x\)
• marks location of “virtual heads”

Repeat:
• scan tape, remembering the symbols under each virtual head in the state
  (how many new states needed?)
• make changes to reflect 1 step of \(M\)
  • if hit \#, shift to right to make room
  if \(M\) halts, erase all but 1st string
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:
    $\delta: Q \times \Gamma \rightarrow 2^Q \times \{L, R\}$

NTM acceptance

- Start configuration: $q_0w$ (w is input)
- Accepting config.: any config. with state $q_{\text{accept}}$
- Rejecting config.: any config. with state $q_{\text{reject}}$

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

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
Nondeterministic TMs

Simulating NTM $M$ with a deterministic TM:

- **STEP 2**: simulate $M$ using string on tape 3 to determine which choice to take at each step
  - if encounter blank, or a $#$ larger than the number of choices available at this step, abort, go to STEP 3
  - if get to a rejecting configuration: DONE = 0, go to STEP 3
  - if get to an accepting configuration, ACCEPT
- **STEP 3**: replace tape 3 with lexicographically next string and go to STEP 2
  - if string lengthened and DONE = 1 REJECT; else DONE = 1

Examples of basic operations

- Convince yourself that the following types of operations are easy to implement as part of TM "program"
  - copying
  - moving
  - incrementing/decrementing
  - arithmetic operations $+$, $-$, $\ast$, $/$

Universal TMs and encoding

- the input to a TM is always a string in $\Sigma^*$
- often we want to interpret the input as representing another object
- examples:
  - tuple of strings $(x, y, z)$
  - $0/1$ matrix
  - graph in adjacency-list format
  - Context-Free Grammar

Universal TMs and encoding

- the input to a TM is always a string in $\Sigma^*$
- we must encode our input as such a string
- examples:
  - tuples separated by $#$: $#x#y#z$
  - $0/1$ matrix given by: $#n#x#$ where $x \in \{0,1\}^n$
- any reasonable encoding is OK
- emphasize "encoding of $X$" by writing $<X>$