What’s on our platter for today?

- An example of a decidable language that is not a CFL
  - Implementation-level description of a TM
  - State diagram of TM
- Varieties of TMs
  - Multi-Tape TMs
  - Nondeterministic TMs
  - String Enumerators
- Closure properties
- Church-Turing Thesis:
  “Algorithm” = Turing Machine

This will ‘rap up Chap 3

Example of a non-CF decidable language

- We know \( L = \{0^n1^n0^n \mid n \geq 0 \} \) is not a CFL (pumping lemma)
- Show \( L \) is decidable
  - Construct a decider \( M \) such that \( L(M) = L \)
  - A decider is a TM that always halts (in \( q_{\text{acc}} \) or \( q_{\text{rej}} \)) and is guaranteed not to go into an infinite loop for any input
Initial Idea for a Decider for \( \{0^n1^n0^n \mid n \geq 0\} \)

✦ General Idea: Match each 0 with a 1 and a 0 following the 1.
✦ Implementation Level Description of a Decider for L:

On input w:
1. If first symbol = blank, ACCEPT
2. If first symbol = 1, REJECT
3. If first symbol = 0, Write a blank to mark left end of tape
   a. If current symbol is 0 or X, skip until it is 1. REJECT if blank.
   b. Write X over 1. Skip 1’s/X’s until you see 0. REJECT if blank.
   c. Write X over 0. Move back to left end of tape.
4. At left end: Skip X’s until:
   a. You see 0: Write X over 0 and GOTO 3a
   b. You see 1: REJECT
   c. You see a blank space: ACCEPT

Seems okay…

Try running the decider on:
- 010, 001100, … ➔ ACCEPT
- 0, 000, 0100, … ➔ REJECT
BUT…

Houston, we have a problem with our Turing machine…

What’s the problem?

✦ Try running the decider on:
  010010, 010001100 \rightarrow \text{ACCEPT}!!!
  Need to fix it…

Maybe it’s that GOTO?
An Aside: Dijkstra on GOTOs

“For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of go to statements in the programs they produce.”


A Simple Fix (to the Decider)

- Scan initially to make sure string is of the form 0*1*0*
- On input w:
  1. If first symbol = blank, ACCEPT
  2. If first symbol = 1, REJECT
  3. If first symbol = 0: if w is not in 00*11*00*, REJECT; else, Write a blank to mark left end of tape
     a. If current symbol is 0 or X, skip until it is 1. REJECT if blank.
     b. Write X over 1. Skip 1’s/X’s until you see 0. REJECT if blank.
     c. Write X over 0. Move back to left end of tape.
  4. At left end: Skip X’s until:
     a. You see 0: Write X over 0 and GOTO 3a
     b. You see 1: REJECT
     c. You see a blank space: ACCEPT
The Decider TM for L in all its glory

Varieties of TMs

What if we allow multiple tapes?

What if we allow nondeterminism?

What if I take nap?
Various Types of TMs

✦ Multi-Tape TMs: TM with k tapes and k heads
  \[ \delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L,R\}^k \]
  \[ \delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, L, R, \ldots, L) \]

✦ Nondeterministic TMs (NTMs)
  \[ \delta : Q \times \Gamma \rightarrow \text{Pow}(Q \times \Gamma \times \{L,R\}) \]
  \[ \delta(q_i, a) = \{(q_1, b, R), (q_2, c, L), \ldots, (q_m, d, R)\} \]

✦ Enumerator TM for L: Prints all strings in L (in any order, possibly with repetitions) and only the strings in L

✦ Other types: TM with Two-way infinite tape, TM with multiple heads on a single tape, 2D infinite tape TM, Random Access Memory (RAM) TM, etc.

Surprise!
All TMs are born equal…

✦ Each of the preceding TMs is equivalent to the standard TM
  \[ \Rightarrow \text{They recognize the same set of languages (the Turing-recognizable languages)} \]

✦ Proof idea: Simulate the “deviant” TM using a standard TM

✦ Example 1: Multi-tape TM on a standard TM
  \[ \Rightarrow \text{Represent } k \text{ tapes sequentially on } 1 \text{ tape using separators } \# \]
  \[ \Rightarrow \text{Use new symbol } a \text{ to denote a head currently on symbol } a \]

\[
\begin{array}{c}
0 1 \ldots \ldots \ldots \ldots \\
\uparrow \\
b a h \ldots \ldots \ldots \ldots \\
\uparrow \\
3 2 2 \ldots \ldots \ldots \\
\uparrow
\end{array} \equiv 
\begin{array}{c}
\# 0 1 \# b a h \# 3 2 2 \# \ldots \ldots \\
\uparrow
\end{array}
\]

(See text for details)
Simulating Nondeterminism

- Any nondeterministic TM $N$ can be simulated by a deterministic TM $M$.
- $N$ accepts $w$ iff there is at least 1 path in $N$’s tree for $w$ ending in $q_{\text{ACC}}$.
- General proof idea: Simulate each branch sequentially.
- Proof idea 1: Use depth first search?
  - No, might go deep into an infinite branch and never explore other branches!
- Proof idea 2: Use breadth first search
  - Explore all branches at depth $n$ before $n+1$.

Simulating Nondeterminism: Details, Details

- Use a 3-tape DTM $M$ for breadth-first traversal of $N$’s tree on $w$:
  - Tape 1 keeps the input string $w$.
  - Tape 2 stores $N$’s tape during simulation along 1 path (given by tape 3) up to a particular depth, starting with $w$.
  - Tape 3 stores current path number.
    - E.g. $\varepsilon = \text{root node } q_0$.
    - $213 = \text{path made up of 3rd child of 1st child of 2nd child of root}$.
- See text for more details.
Closure Properties of Decidable Languages

- Decidable languages are closed under $\cup$, $^\circ$, $\ast$, $\cap$, and complement

- Example: Closure under $\cup$

- Need to show that union of 2 decidable L’s is also decidable

  Let $M_1$ be a decider for $L_1$ and $M_2$ a decider for $L_2$

  A decider $M$ for $L_1 \cup L_2$:

  On input $w$:

  1. Simulate $M_1$ on $w$. If $M_1$ accepts, then ACCEPT $w$. Otherwise, go to step 2 (because $M_1$ has halted and rejected $w$)

  2. Simulate $M_2$ on $w$. If $M_2$ accepts, ACCEPT $w$ else REJECT $w$.

  $M$ accepts $w$ iff $M_1$ accepts $w$ OR $M_2$ accepts $w$

  i.e. $L(M) = L_1 \cup L_2$

Closure Properties of Decidable Languages

- Example: Closure under $\cup$

  Let $M_1$ be a decider for $L_1$ and $M_2$ a decider for $L_2$

  A decider $M$ for $L_1 \cup L_2$:

  On input $w$:

  1. Simulate $M_1$ on $w$. If $M_1$ accepts, then ACCEPT $w$. Otherwise, go to step 2 (because $M_1$ has halted and rejected $w$)

  2. Simulate $M_2$ on $w$. If $M_2$ accepts, ACCEPT $w$ else REJECT $w$.

  $M$ accepts $w$ iff $M_1$ accepts $w$ OR $M_2$ accepts $w$

  i.e. $L(M) = L_1 \cup L_2$

Will this proof work for showing Turing-recognizable languages are closed under $\cup$? Why/Why not?

Uh…I dunno.

Wait, will $M_1$ always halt?!
Closure for Recognizable Languages

✦ Turing-Recognizable languages are closed under $\cup$, $\circ$, $\ast$, and $\cap$ (but not complement! We will see this later in Chapter 4)

✦ Example: Closure under $\cap$
   Let $M_1$ be a TM for $L_1$ and $M_2$ a TM for $L_2$ (both may loop)
   A TM $M$ for $L_1 \cap L_2$:
   On input $w$:
   1. Simulate $M_1$ on $w$. If $M_1$ halts and accepts $w$, go to step 2. If $M_1$ halts and rejects $w$, then REJECT $w$. (If $M_1$ loops, then $M$ will also loop and thus reject $w$)
   2. Simulate $M_2$ on $w$. If $M_2$ halts and accepts, ACCEPT $w$. If $M_2$ halts and rejects, then REJECT $w$. (If $M_2$ loops, then $M$ will also loop and thus reject $w$)
   $M$ accepts $w$ iff $M_1$ accepts $w$ AND $M_2$ accepts $w$ i.e. $L(M) = L_1 \cap L_2$

R. Rao, CSE 322

That wraps up Chapter 3!
Next 2 Classes: Undecidable Problems
Now: Fill out student evals

Look, Ma, I’m on CSE 322!

Always thought he was nuts…