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

Example of a non-CF decidable language

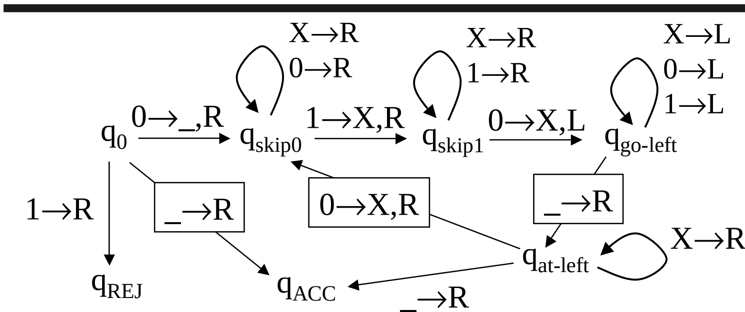
- We know $L = \{0^n1^n0^n \mid n \geq 0\}$ is not a CFL (pumping lemma)
- Show L is decidable
  - Construct a decider M such that L(M) = L
  - A decider is a TM that always halts (in q_{acc} or q_{rej}) and is guaranteed not to go into an infinite loop for any input

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

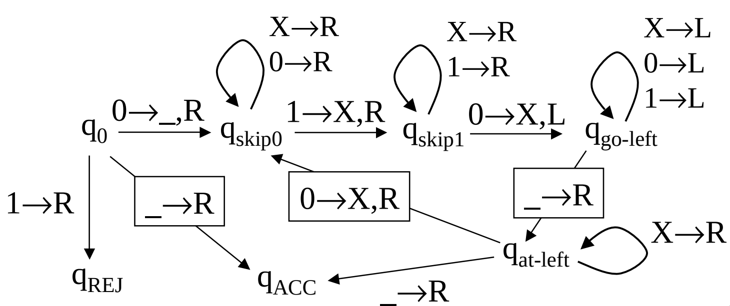
Seems okay…



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

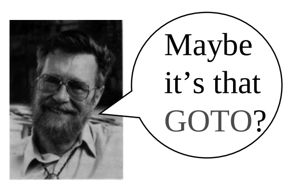
BUT…

What's the problem?



- Try running the decider on:
  - 010010, 010001100 → ACCEPT!!!

Need to fix it…



E. W. Dijkstra

An Aside: Dijsktra 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."

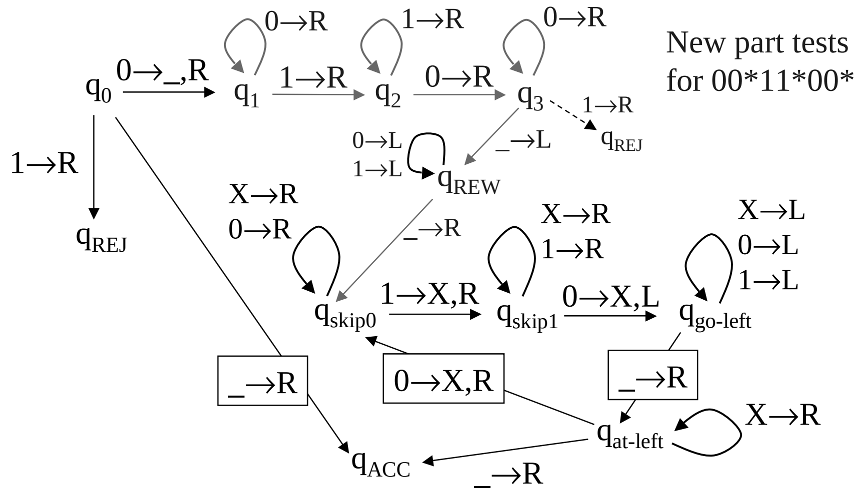
Opening sentence of: "Go To Statement Considered Harmful" by Edsger W. Dijkstra, Letter to the Editor, Communications of the ACM, Vol. 11, No. 3, March 1968, pp. 147-148.

A Simple Fix (to the Decider)

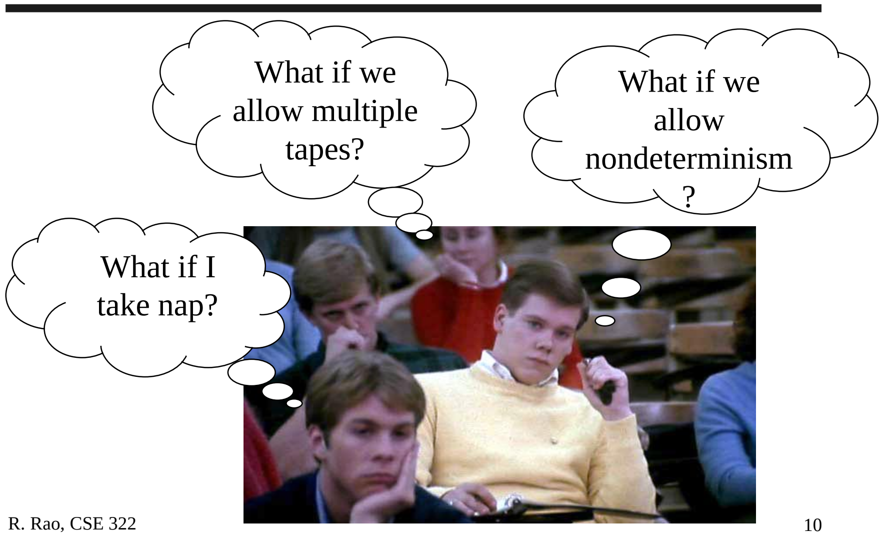
- Scan initially to make sure string is of the form 0*1*0*
- On input w:
  1. If first symbol = blank, ACCEPT
  2. If first symbol = 1, REJECT
  3. If first symbol = 0: if w is not in 00*11*00*, REJECT; else, Write a blank to mark left end of tape
     a. If current symbol is 0 or X, skip until it is 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

Add this

The Decider TM for L in all its glory



Varieties of TMs



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)
  - $\delta: Q \times \Gamma \rightarrow Pow(Q \times \Gamma \times \{L,R\})$
  - $\delta(q_i, a) = \{(q_1, b, R), (q_2, c, L), \ldots, (q_m, d, R)\}$
- 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
  - 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
  - Represent k tapes sequentially on 1 tape using separators #
  - Use new symbol $\underline{a}$ to denote a head currently on symbol *a*

0 1 . . . .

b a h . . . .

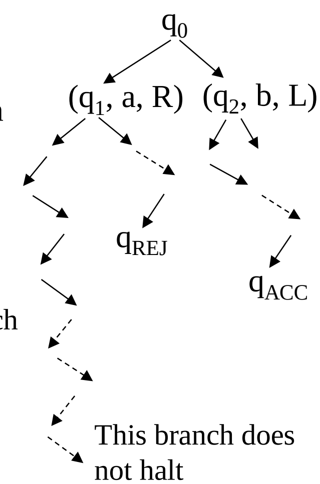
3 2 2 . . . .

$\equiv$ # 0 $\underline{1}$ # b $\underline{a}$ h # $\underline{3}$ 2 2 # . . . .

(See text for details)

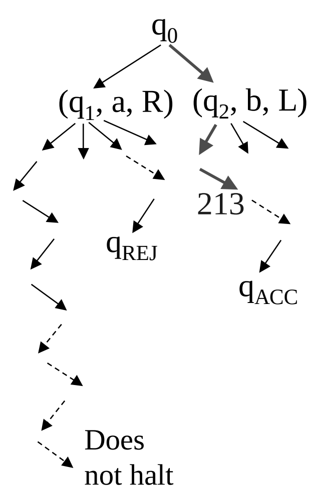
Simulating Nondeterminism

- 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_{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. ε = root node q_0
    213 = 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 $\cup$, $\circ$, $*$, $\cap$, and complement
- Example: Closure under $\cup$
- Need to show that union of 2 decidable L's is also decidable

  Let M1 be a decider for L1 and M2 a decider for L2

  A decider M for L1 $\cup$ L2:

  On input w:

  1. Simulate M1 on w. If M1 accepts, then ACCEPT w. Otherwise, go to step 2 (because M1 has halted and rejected w)
  2. Simulate M2 on w. If M2 accepts, ACCEPT w else REJECT w.

  M accepts w iff M1 accepts w OR M2 accepts w

  i.e. L(M) = L1 $\cup$ L2

Closure Properties of Decidable Languages

- Example: Closure under $\cup$

  Let M1 be a decider for L1 and M2 a decider for L2

  A decider M for L1 $\cup$ L2:

  On input w:

  1. Simulate M1 on w. If M1 accepts, then ACCEPT w. Otherwise, go to step 2 (because M1 has halted and rejected w)
  2. Simulate M2 on w. If M2 accepts, ACCEPT w else REJECT w.

  M accepts w iff M1 accepts w OR M2 accepts w

  i.e. L(M) = L1 $\cup$ L2

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

Closure for Recognizable Languages

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

  Let M1 be a TM for L1 and M2 a TM for L2 (both may loop)

  A TM M for L1 $\cap$ L2:

  On input w:

  1. Simulate M1 on w. If M1 halts and accepts w, go to step 2. If M1 halts and rejects w, then REJECT w. (If M1 loops, then M will also loop and thus reject w)
  2. Simulate M2 on w. If M2 halts and accepts, ACCEPT w. If M2 halts and rejects, then REJECT w. (If M2 loops, then M will also loop and thus reject w)

  M accepts w iff M1 accepts w AND M2 accepts w i.e. L(M) = L1$\cap$L2

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

Always thought he was nuts…

Look, Ma, I'm on CSE 322!