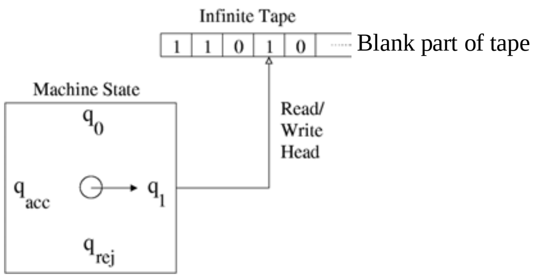
Turing Machines Review

- 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
- Church-Turing Thesis:
  "Algorithm" ≡ Turing Machine

Turing Machines



Just like a DFA except:

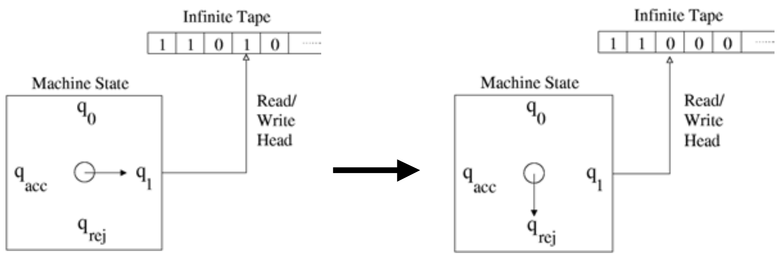
- You have an infinite "tape" memory (or scratchpad) on which you receive your input and on which you can do your calculations
- You can <u>read</u> one symbol at a time from a cell on the tape, <u>write</u> one symbol, then <u>move</u> the read/write pointer (head) left (L) or right (R)

Who's Turing?

- Alan Turing (1912-1954): one of the most brilliant mathematicians of the 20th century (one of the "founding fathers" of computing)
- Click on "Theory Hall of Fame" link on class web under "Lectures"
- Introduced the Turing machine as a formal model of what it means to compute and solve a problem (i.e. an "algorithm")
  - Paper: `On computable numbers, with an application to the Entscheidungsproblem, Proc. London Math. Soc. 42 (1936).`

How do Turing Machines compute?

- δ(current state, symbol under the head) = (next state, symbol to write over current symbol, direction of head movement)



- Diagram shows: **$\delta(q_1,1) = (q_{rej}, 0, L)$** (R = right, L = left)
- In terms of "Configurations": $110q_1 10 \Rightarrow 11q_{rej}000$

Solving Problems with Turing Machines

- 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*

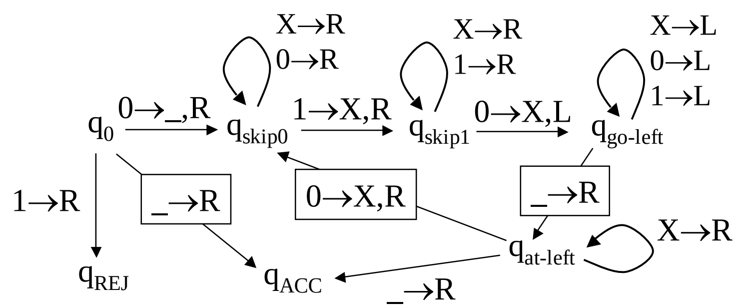
Idea for a Decider for $\{0^n1^n0^n \mid n \geq 0\}$

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

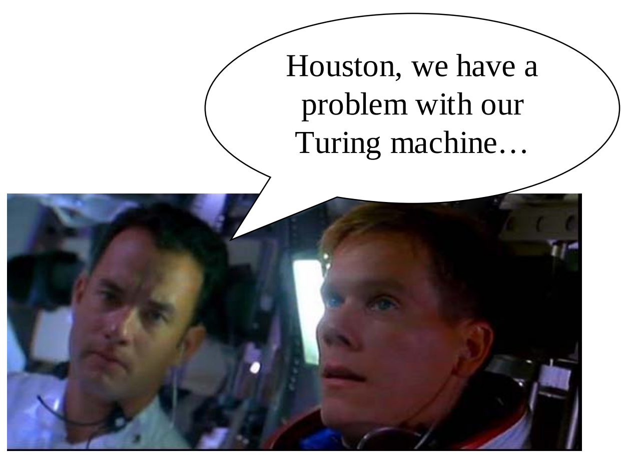
On input w:

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

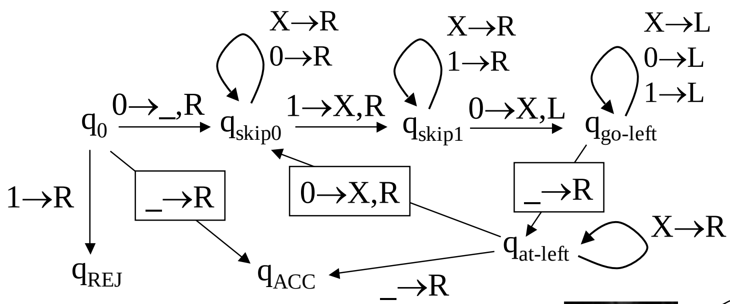
State Diagram



- Try running the decider on:
  - 010, 001100, … → ACCEPT
  - 0, 000, 0100, … → REJECT
  - What about 010010?



What's the problem?



- The decider accepts incorrect strings:
  - 010010, 010001100 → ACCEPT!!!
  - Accepts $(0^n1^n0^n)^*$

E. W. Dijkstra

Need to fix it…

An Aside: Dijsktra on GOTOs

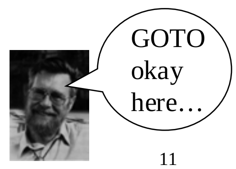
"For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of go to statements in the programs they produce."

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.

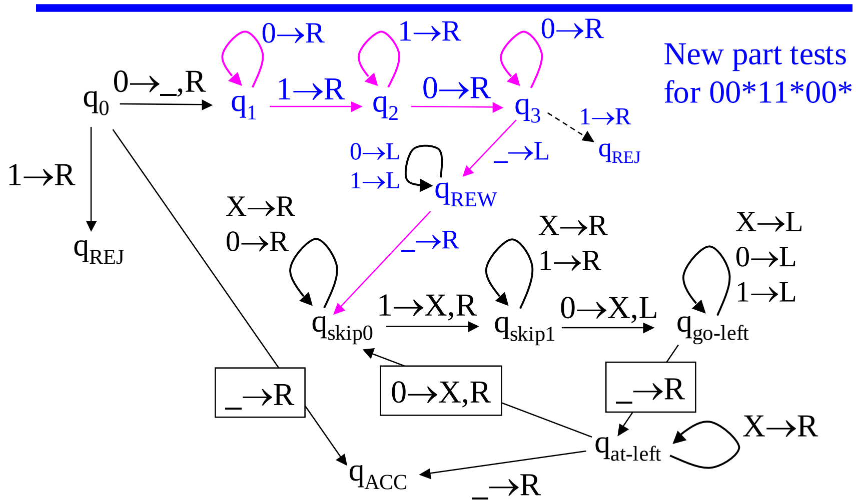
A Simple Fix (to the Decider)

- 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

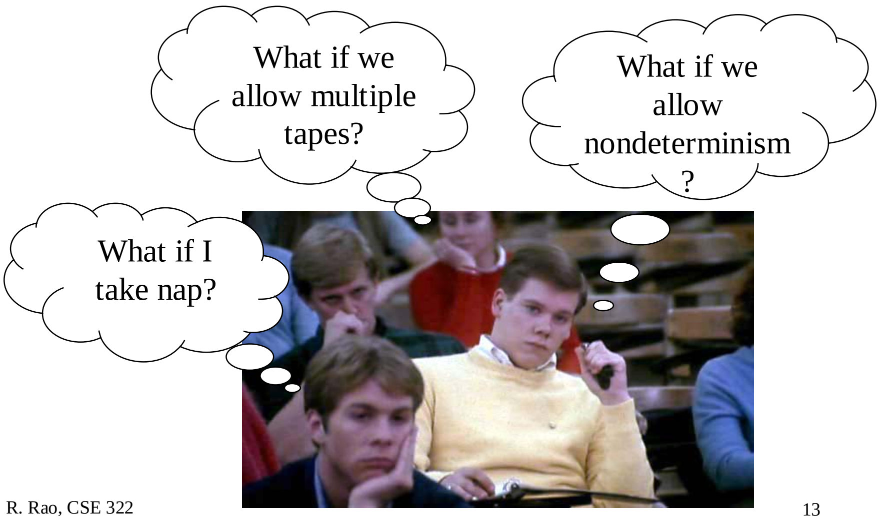
Add this



The Decider TM for L in all its glory



Varieties of TMs

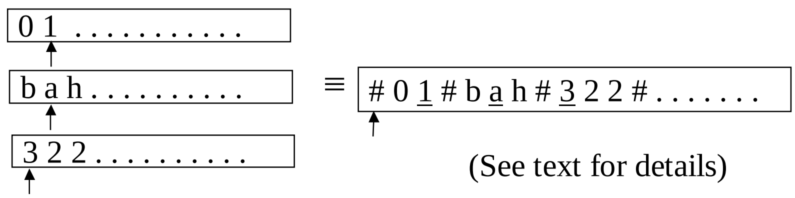


Various Types of TMs

- **Multi-Tape TMs**: TM with k tapes and k heads
  - δ: $Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L,R\}^k$
  - $\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, L, R, \ldots, L)$
- **Nondeterministic TMs (NTMs)**
  - δ: $Q \times \Gamma \rightarrow Pow(Q \times \Gamma \times \{L,R\})$
  - $\delta(q_i, a) = \{(q_1, b, R), (q_2, c, L), \ldots, (q_m, d, R)\}$
- **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…

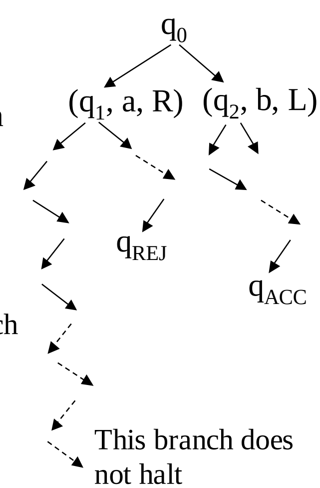
- Each of the preceding TMs is equivalent to the standard TM
  - They recognize the same set of languages (the Turing-recognizable languages)
- Proof idea: Simulate the "deviant" TM using a standard TM
- Example 1: Multi-tape TM on a standard TM
  - Represent k tapes sequentially on 1 tape using separators #
  - Use new symbol *a̲* to denote a head currently on symbol *a*



(See text for details)

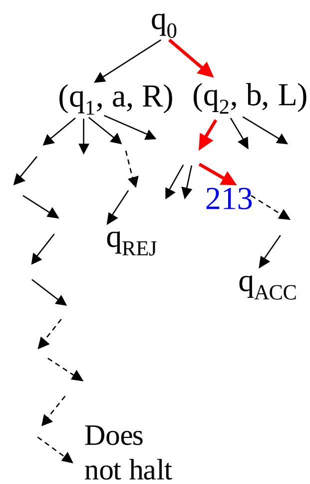
Example 2: Simulating Nondeterminism

- 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

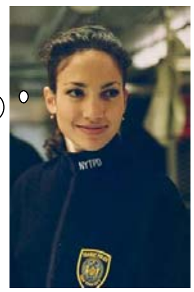
- Use a 3-tape DTM M for breadth-first traversal of N’s tree on w:
  - Tape 1 keeps the input string w
  - Tape 2 stores N’s tape during simulation along 1 path (given by tape 3) up to a particular depth, starting with w
  - Tape 3 stores current path number
    E.g. ε = root node q_0
    213 = path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



The Church-Turing Thesis

- Various definitions of “algorithms” were shown to be equivalent in the 1930s
- **Church-Turing Thesis**: “The intuitive notion of algorithms equals Turing machine algorithms”
  - Turing machines serve as a precise formal model for the intuitive notion of an algorithm
- “Any computation on a digital computer is equivalent to computation in a Turing machine”

Next Class: We'll 'rap up Chapta 3.
Now: Ya CSE homiez fill out da student evals



Look, Ma, I'm on CSE 322!

That guy needs help…

Have a great weekend!