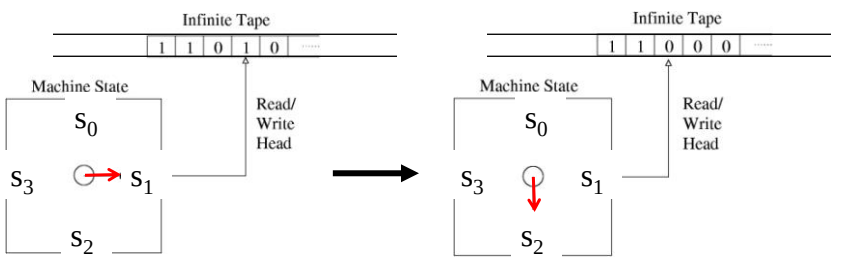
What we will munch on today…

- Turing Machines
- Church-Turing Thesis
- Unsolvable Problems
- Sections 12.5 and 3.1 in the text

Guest appearance

How do Turing Machines compute?

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



- 5-tuple representation: **(s_1, 1, s_2, 0, L)** (R = right, L = left)
- Turing machine "program" = set of such 5-tuples

Turing Machine (TM) Definition

- TM T = (S, V, I, f, s_0, F)
  - NOTE: We will use V and F in our definition of TMs; the textbook does not. Using V makes the input alphabet clear and distinct from tape alphabet I. Using F makes the final/accepting states clear.
- S, s_0, F are as in DFA definition
- Input strings are over an alphabet V $\subseteq$ I.
  - TM can use other symbols in I as markers, etc. for computing.
  - Blank symbol □ is always in I (and not in V).
- f maps (state1, symbol1) to (state2, symbol2, direction)
  - f need not be defined for every (state,symbol) input
    - f is a "partial function"
  - If f not defined for a particular (state,symbol), TM halts.

Turing Machine (TM) Details

- Input string to TM given on tape
  - TM always starts on leftmost nonblank symbol
  - If no input, then can start on any cell of the tape
- TM can halt in two types of states:
  - TM halts and accepts the input iff it enters a final state in F
  - TM halts and rejects the input when it halts in any other state (when f is not defined for a (state, symbol) pair)
- TM *recognizes* a string *w* iff it halts in a final state for *w*
  - TM can reject *w* by halting in any non-final state or by looping forever!

Solving Problems with Turing Machines

- TM that recognizes L = $\{a^n b^n c^n \mid n \geq 0\}$ and always halts
  - Such TMs are called decider TMs.

Input: aaabbbccc

Idea: Mark off each a, b, and c with x, y, z. Accept if each a, b, c could be matched, and only x's,y's,z's remain

aaabbbccc
xaabbbccc
xaaybbccc
xaaybbzcc
xxaybbzcc
….
xxxyyyzzz

TM for $\{a^n b^n c^n \mid n \geq 0\}$

- Implementation Level Description of the TM:

On input w:

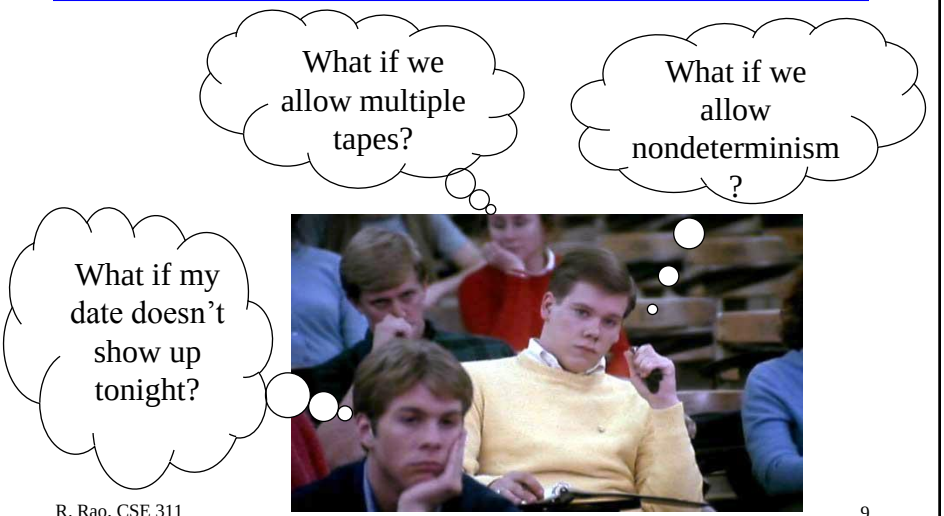
1. If first symbol = blank, ACCEPT
2. If first symbol = b or c, REJECT
3. If first symbol = a,
   a. Write X over a and move right.
   b. If current symbol is a or y, move right until you see b. REJECT if other symbol.
   c. Write y over b. Skip b's and z's until you see c. REJECT if other symbol.
   d. Write z over c. Rewind back to rightmost x.
4. If you see an a, go to 3a. If you see y, rewind and check if the tape only has x's, y's, z's. If so, ACCEPT, otherwise REJECT.

jFLAP demo

- To run the demo, download and run in jFLAP the file turingAnBnCn.jff at:
  http://www.cs.duke.edu/csed/jflap/tutorial/turing/one/turingAnBnCn.jff

Can we augment the power of Turing machines with various accessories?

Varieties of TMs

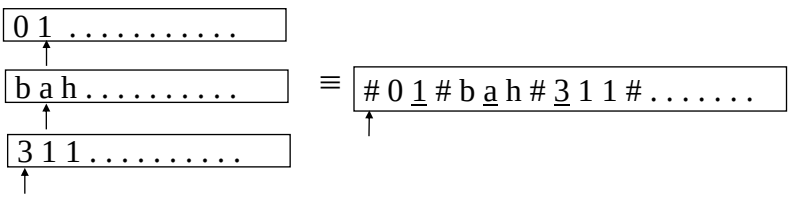


Various Types of TMs

- **Multi-Tape TMs**: TM with k tapes and k heads
  - f: $S \times I^k \rightarrow S \times I^k \times \{L,R\}^k$
  - $f(s_i, a_1, \ldots, a_k) = (s_j, b_1, \ldots, b_k, L, R, \ldots, L)$
- **Nondeterministic TMs (NTMs)**
  - f: $S \times I \rightarrow Pow(S \times I \times \{L,R\})$
  - $f(s_i, a) = \{(s_1, b, R), (s_2, c, L), \ldots, (s_m, d, R)\}$
- Other types: 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
- Proof idea: Simulate the "deviant" TM using a standard TM
- Example: 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



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"

A language that can be recognized by a decider TM (always halts) is called a *decidable* language

Some decidable languages:

$\{a^n b^n c^n \mid n \geq 0\}$,
1*01*01* (or any reg. exp.),
$\{0^p \} \mid p$ is prime$\}$,
{P | P is a syntactically correct Java program}

Decidable languages correspond to problems that we can solve using an algorithm (no infinite loops!)

Are there languages that are not decidable?

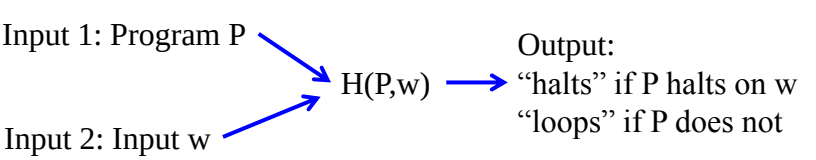
(i.e., from Church-Turing thesis,
are there problems that are not solvable
by any computer program?

The Halting Problem

- Consider problem: Given a program P and input w, does P halt on w?
  - Equivalently, given TM M and input w, does M halt on w?
- Good for debugging CSE 142/143 programs…
- Naïve solution: What if we run P on w and see what happens?

The Halting Problem is Unsolvable

- Theorem: There is no program which, when given as input a program P and its input w, decides whether P halts on w.
- Proof: By contradiction.
- Assume such a program exists – call it H.

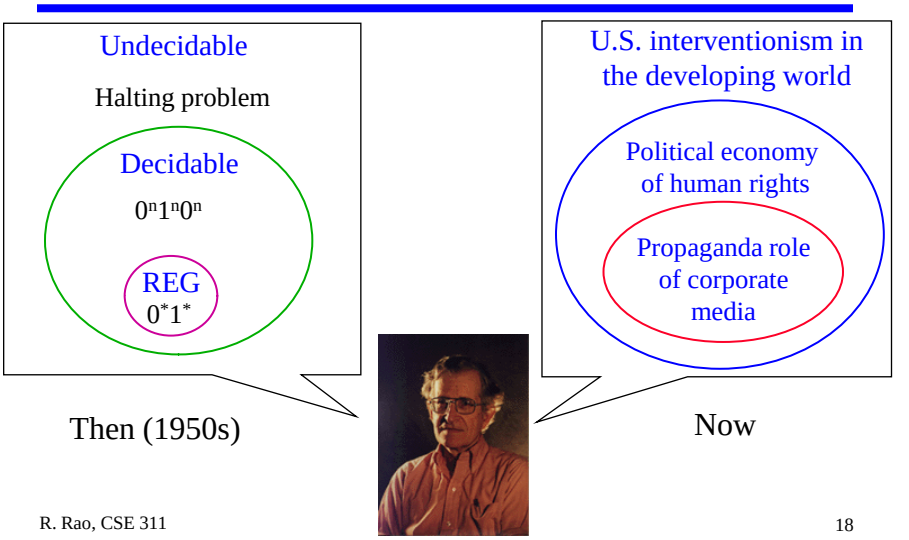


The Halting Problem is Unsolvable

- If H exists, we can create a new program K which takes as input a program P and uses H as a subroutine as follows:
  - On Input P, compute H(P,P).
  - If H(P,P) = "loops", then K(P) halts.
  - If H(P,P) = "halts", then K(P) loops forever.
- Consider what happens when input P is K itself, i.e., K(K):
  K loops on K → H(K,K) = "loops" → K(K) halts (contradiction)
  K halts on K → H(K,K) = "halts" → K(K) loops (contradiction)
  - Both cases give a contradiction
- Therefore, H cannot exist, i.e., halting problem is unsolvable.

  QED

The Chomsky Hierarchy – Then & Now…



Dat raps up Turing Machines…
Let's close with a tribute to
da pumpin' lemma

Da Pumpin' Lemma

(adapted from a poem by Harry Mairson)

Hear it on the new album:
Dig dat funky DFA

Any regulah language L hassa magic numba p
Any long word s in L hasda followin' propa'ty:
In its first p symbols issa segment u can find
Whoz repetition or omission leaves s amongst its kind.

If ya find a langa L which fails dis acid test,
And a long word ya pump becomes distinct from all da rest,
By contradixion ya have shown that L ain't certainly not
A regular homie that is resilient to da pumpin' that u've wrought.

If on the otha' hand, s stays within its L,
Then eitha L is regulah, or else ya chose not well.
s is xyz y'all where y is not empty,
And y must come befo' da $p+1^{st}$ symbol ya see.