CSE 431

Lecture 6 Intro to Theory of Computation

So far...

k-tape TM $\equiv$ 1-tape TM

NTM $\equiv$ TM

2-dim TM $\equiv$ TM (Homework)

Also TM $\equiv$ Random Access Machine

Random Access Machine

Idealized computer
- Just like model for ordinary assembly language except that
  - each register holds an arbitrary integer (initially 0)
  - there is one register for each natural number 0, 1, 2, ...
- Indirect addressing still allowed.
  - register indexed uses absolute value

Simulation: Configuration - store value of each register touched between # ~ # on tape

e.g. # $\underset{\text{binary}}{i}$ # $\underset{\text{value}}{R_i}$ # (rest are implicitly 0)

$\Rightarrow$ C, Java, any programming language + unbounded memory $\equiv$ TM

Also $\underbrace{\text{TM} \equiv}_{\text{Turing 1936}}$ λ-calculus $\underbrace{\equiv \mu}_{\text{Kleene 1937}}$-recursive functions

Church-Turing Thesis (1936)

$$\boxed{\text{Any reasonable model that captures all of computation is equivalent in power to Turing machines}}$$

Not a statement that can be proved or disproved since it uses a notion "reasonable model" which is not formal

The text phrases this as

$$\boxed{\text{Algorithm} \equiv \text{TM Algorithm}}$$

but this only makes sense if the left side is meant to say "our intuitive notion of algorithm"

Since it was put forth, many other models of computation have been developed and all models are either

- equivalent to TM in power (or weaker) e.g. quantum computers
- allow unreasonable operations with infinite action in a single step

Such models sometimes called "Turing-complete"

From now on we mostly use high-level descriptions just as in ordinary pseudocode
- not worrying about direction of head movements etc.

Input encoding:

many problems have multiple input encodings

eg. Graphs $G=(V,E)$
- adjacency matrix
- adjacency lists (eg. singly, doubly linked)
- edge lists

a TM can convert any one of these encodings to the other so we don't care which one.

This is a string over some fixed alphabet

Use $\langle G\rangle$ to denote any reasonable encoding

angle brackets in latex `$\langle G\rangle$`

$$\{\langle G\rangle \mid G \text{ is a graph and } \dots\}$$

Describe TM operating on such an input like any high level graph algorithm.

More generally, can encode many things between angle brackets.

eg. $\{\langle G,s,t\rangle : G \text{ is a graph with a path from node } s \text{ to node } t\}$

or $\{\langle x,y\rangle : x, y \in \Sigma^*\}$ encodes a pair of strings

e.g. Input is a multivariate polynomial p
say
$$p = 6x^2y + 3yz + 3$$ in vars x, y, z.

$\langle p\rangle$: list of coefficients and monomials

Can also use $\langle M\rangle$ to denote encodings of computational devices
- e.g. finite state machines
  $\underbrace{DFA}$, $\underbrace{NFA}$, ...
  labelled graphs.

e.g. define $E_{DFA} = \{\langle M\rangle : M$ is a DFA with $L(M) = \emptyset\}$

Recall a DFA M has

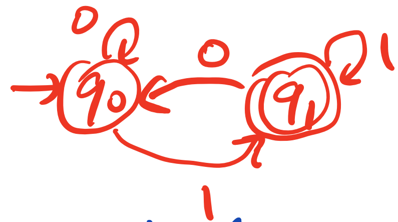
Q, Σ, δ, $q_0 \in Q$, $F \subseteq Q$

set of states, set of symbols, transition function, start states, set of accepting / final state

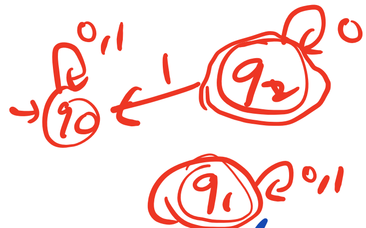
δ	Σ
Q	table

- encoding is similar to that of graphs
- must encode each symbol in Σ since $\langle M\rangle$ is over a fixed alphabet.

eg.



$L(M) \neq \emptyset$



$L(M) = \emptyset$

Thm E_{DFA} is decidable

Proof For a DFA, $L(M)$ is empty iff no path from the start state q_0 to any of the states in F

Algorithm: Do a graph search (DFS, BFS, ...) in diagram of M starting at q_0
- if state in F reached, reject
- if no state in F reached accept

∎

$E_{NFA} = \{\langle M \rangle : M \text{ is an NFA and } L(M) = \emptyset\}$

Thm E_{NFA} is decidable

Proof Algorithm 1:
- Convert $\langle M \rangle$ to $\langle M' \rangle$ where M' is a DFA with $L(M') = L(M)$ using subset construction
- Run decider for E_{DFA} on input $\langle M' \rangle$

Algorithm 2: (Same as algorithm for E_{DFA} on diagram of M

check if state of F reachable from q_0

∎

$A_{DFA} = \{\langle M, w\rangle : M$ is a DFA that accepts string $w\}$

Thm A_{DFA} is decidable

Proof: TM

Can do a step-by-step simulation of M on input w.

- keep pointer on next char to be read (on tape)
- record current state of M (on tape)

need this since M might have more states than the TM

Accept if state in F reached at end of w ∎

$A_{NFA} = \{\langle M, w\rangle :$ NFA M accepts $w\}$

Thm A_{NFA} is decidable

Proof

Algorithm I:
- Convert input $\langle M, w\rangle$ for NFA M to $\langle M', w\rangle$ for equivalent DFA M'
- Run decider for A_{DFA} on $\langle M', w\rangle$

Algorithm 2: • Do an "on-the-fly" simulation keeping track of all states reachable from q_0 after reading each character of w. ∎

$A_{REG} = \{\langle R, w\rangle :$ R is a regular expression that generates $w\}$

Thm A_{REG} is decidable

Proof On input $\langle R, w\rangle$ convert R to an equivalent NFA M and run decider for A_{NFA} on input $\langle M, w\rangle$

∎

$EQ_{DFA} = \{\langle M_1, M_2\rangle \mid M_1$ and M_2 are DFAs with $L(M_1) = L(M_2)\}$

Thm EQ_{DFA} is decidable

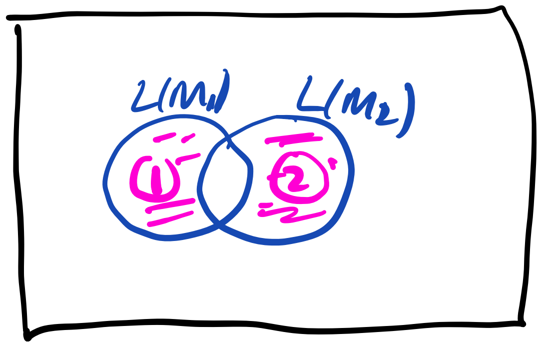
Proof We give two algorithms based on
1. closure properties of DFAs
2. minimization alg for DFAs

Closure

Recall from 311

Given DFAs M, M' we can build new DFA for

- complement $\overline{L(M)}$
  change F to $Q-F$
- $L(M) \cup L(M')$
  $L(M) \cap L(M')$ } cross-product constructions



$L(M_1) = L(M_2)$ iff ① $\cup$ ② is empty.

① $= L(M_1) \cap \overline{L(M_2)}$
② $= L(M_2) \cap \overline{L(M_1)}$

Algorithm 1: Build DFA $\langle M \rangle$ s.t.

$$L(M) = \left(L(M_1) \cap L(\overline{M_2})\right) \cup \left(L(M_2) \cap L(\overline{M_1})\right)$$

- Run decider for E_{DFA} on input $\langle M \rangle$ and output its answer

---

Minimization: Recall DFA minimization alg from 311. We didn't prove it but it turns out that if $L(M_1) = L(M_2)$

Then the minimized versions of M_1 and M_2 will be the same up to renaming of states (which is easy to check since edge labels need to match)

Algorithm 2: • Run minimization alg on M_1 and on M_2

• Check that minimized forms are the same (up to state renaming)

Note: This second alg is efficient in practice (much more than the first)

• In fact this second algorithm was what was used to grade your CSE 311 finite state machine homework!

$\blacksquare$