Lecture 30

Review & Wrapup

Computability Theory

See Midterm Review Slides

From Lec 14

Real Computers are *Finite*

Unbounded "memory" is critical to most undecidability pfs

Real computers are finite: n bits of state (registers, cache, RAM, HD, …) $\Rightarrow \leq 2^n$ configs – it's a DFA!

"Does M accept w" is *decidable*: run M on w; if it runs more that 2^n steps, it's looping. (Recall LBA pfs.)

BUT:

- 2^n is *astronomical:* a modest laptop has n = 100's of gigabits of state; # atoms in the universe ~ 2^{262}

From Lec 14

Are "real" computer problems undecidable?

Options:

- 100 G is so much >> 262, let's say it's approximately unbounded ⇒ undecidable
- Explore/quantify the "computational difficulty" of solving the (decidable) "bounded memory" problem

1st is somewhat crude, but easy, and not crazy, given that we really don't have methods that are fundamentally better for 100Gb memories than for arbitrary algorithms

2nd is more refined but harder; goal of next few weeks is to develop theory supporting such aims

Time & Space Complexity

Defined on TM's but largely model-independent

(1-tape, multi-tape, RAMs, …)

Esp. if we focus on *asymptotic* complexity, *up to polynomials*

E.g. P, PSPACE

For *space*, model-independence even extends to *nondeterministic* models

For *time*, this is a major open problem

E.g., does P = NP?

P

Many important problems are in P: solvable in deterministic polynomial time

- Details are more the fodder of algorithms courses, but we've seen a few examples here, plus many other examples in other courses

Few problems *not* in P are routinely solved;

- For those that are, practice is usually restricted to small instances, or we're forced to settle for approximate, suboptimal, or heuristic "solutions"

A major goal of complexity theory is to delineate the boundaries of what we can feasibly solve

NP

The tip-of-the-iceberg in terms of problems conjectured not to be in P, but a very important tip, because

a) they're very commonly encountered, probably because

b) they arise naturally from basic "search" and "optimization" questions.

Definition: poly time NTM

Equivalent views: poly time verifiable, "guess and check", "is there a…" – all useful

NP-completeness

Defn & Properties of $\leq_p$

A is NP-hard: everything in NP reducible to A

A is NP-complete: NP-hard and *in* NP

  "the hardest problems in NP"

  "All alike under the skin"

Most known natural problems in NP are complete

  #1: 3CNF-SAT

  Many others: Clique, VertexCover, HamPath, Circuit-SAT,…

Beyond NP

"Polynomial Hierarchy":

- Quantified Boolean formulas with fixed number of alternations of $\exists$, $\forall$
- Collapses if NP = co-NP
- Important in helping recognize variants of NP problems

PSPACE

Exponential Time

Double-Exponential Time

…

Complexity class relationships

$P \subseteq NP \cap \text{co-}NP \subseteq NP \cup \text{co-}NP \subseteq PSPACE \subseteq ExpTime$

$NP \neq \text{co-}NP$?

All containments above proper ?

A taste of things we didn't get to

Resource-bounded Hierarchy Theorems:

If $t(n) << T(n)$ (e.g., $\lim_{n\to\infty} t(n)/T(n) = 0$), then

$DSPACE(t(n)) \subsetneq DSPACE(T(n))$

Similar for DTIME, (but fussier about "<<")

E.g.: $TIME(n) \subsetneq TIME(n^2) \subsetneq TIME(n^3) \ldots$

$P \subsetneq TIME(2^n) \subsetneq TIME(3^n) \subsetneq \ldots TIME(2^{n^2}) \subsetneq TIME(2^{2^n})$

Method: diagonalization again

NSPACE is closed under complementation

Is there an s-t path in G?

Is there *no* s-t path in G?

Final Exam

Monday, 2:30

In this Classroom

Two pages of notes allowed; otherwise closed book.

Coverage: comprehensive

- Sipser, Chapters 3, 4, 5; 7, 8.1-8.3
- Lectures
- Homework

Some bias (~ 60/40) towards topics since midterm

Thanks, and Good Luck!