Harvard CS 121 and CSCI E-207
Lecture 23: Conclusions

Harry Lewis

December 2, 2010

End of Term Miscellany

- PS9 due Tuesday, December 7—NO LATE PSETS ACCEPTED
- CS 121 Review: Wednesday December 15 5:30–7:30pm, MD G125 (videorecorded)
- CSCI E-207: Consult Deborah
- We will email you our records of your grades before the exam, please point out any discrepancies within 24 hours

Final Examination

- CS 121 exam Monday, Dec. 20, 9am, Science Center E (3 hours)
- CSCI E-207 students will hear directly from Deborah
- Closed book
- Old exams will be posted for practice, solutions are subject of CS 121 review

Space complexity

- "Space" as a resource $=$ number of TM squares used in a computation
- A reasonable proxy for memory on other computational models
- PSPACE $=$ languages decidable on TMs using polynomial space
- P $\subseteq$ PSPACE (why?)
- NP $\subseteq$ PSPACE (why?)
- PSPACE $\subseteq$ NPSPACE (why?)

Savitch's Theorem

NPSPACE $=$ PSPACE

- How much time can a computation take if it uses $O(n^k)$ space and does not loop? $O(2^{n^k})$

- To check deterministically if there exists a computation from TM configuration C_1 to TM configuration C_2 in T steps,
  - for all configurations C, check if
    - there is a computation from C_1 to C of $T/2$ steps and
    - there is a computation from C to C_2 in $T/2$ steps

- To check whether initial configuration yields final configuration takes $O(\log(2^{n^k})) = O(n^k)$ recursion levels and $O(n^k)$ space at each level $\Rightarrow O(n^{2k})$ space

A problem in PSPACE but probably not in NP

- Determining whether a quantified boolean expression is true is in PSPACE
  - E.g. $\forall x \exists y \forall z [(x \wedge y) \vee (\neg x \wedge \neg z)]$
- Why?
- In fact this problem is complete for PSPACE
- Proof:

LOGSPACE

A problem is in LOGSPACE if it is decided by a TM that uses only logarithmic space on a work tape and doesn't write on the input tape.

$$\text{LOGSPACE} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NPSPACE}$$

LOGSPACE $\subsetneq$ PSPACE by a diagonalization argument

So at least one of the three $\subseteq$s must be a $\subsetneq$ but we don't know which!

Reprise: Models of computation and formal systems

- DFAs, NFAs, REs, CFGs, PDAs, TMs, NTMs,...
- How to to formally model computation
- Asymptotic perspective (fixed program for all input lengths)
- Design your own models as circumstances demand (eg interactive/distributed computation, randomized computation, biological systems, economic systems)

Classification of computational problems

- Positive results: regular, context-free, polynomial-time, decidable, Turing-recognizable languages
- Negative results: non-regular, non-CF, NP-complete(?), undecidable, non-recognizable languages
- Notion of reduction between problems
- The systematic methodology for proving things impossible is one of the most important achievements of computer science.
- NP-completeness is one of the most important "exports" of computer science to the rest of science.

Understanding Intractability

- Many important problems are NP-complete (or even undecidable).
- But also some great positive results in algorithms design
  - E.g. poly-time algs for LINEAR PROGRAMMING, PRIMALITY TESTING, POLYNOMIAL FACTORIZATION, NETWORK FLOWS, ... (take CS124, CS222, CS223, CS225, CS226r)
- What does NP-completeness mean? (assuming P $\neq$ NP)
  - No algorithm can be guaranteed to solve the problem perfectly in polynomial time on <u>all</u> instances
  - Exhaustive search is often unavoidable
  - Mathematical nastiness: no nice, closed form solutions.

Coping with Intractability

- What if you need to solve an NP-complete (or undecidable) problem?
  - Ask your boss for a new assignment. :-)
  - Simplify the problem, you may not need to solve it in full generality
  - Identify <u>additional constraints</u> that make the problem easier (eg bounded-degree graphs, ILP with fixed number of variables, 2-SAT)
  - <u>Approximation algorithms</u>, e.g. find a TSP tour of length at most 1.01 times the shortest.
  - <u>Average-case analysis</u> — analyze running time or correctness on "random" inputs. (Often hard to find distribution that models "real-life" inputs well.)

More attacks on intractable problems

- Heuristics — techniques that seem to work well in practice but do not have rigorous performance guarantees.
- Change the problem
  - Instead of verifying that general programs satisfy desired security properties (undecidable), ask programmers to supply programs with (easily verifiable) “proofs” that the properties are satisfied
  - Change the programming language (CS 152, CS 252r)

Theory of Computation after CS 121

- Algorithms
  - CS 124: Algorithms & Data Structures (every Spring)
  - CS 222: Algorithms at the End of the Wire
  - CS 226r: Efficient Algorithms
- Computational Complexity
  - CS 221: Computational Complexity
  - CS 225: Pseudorandomness

Theory of Computation after CS 121, cont.

- Cryptography
  - CS 120: Introduction to Cryptography

Theory of Computation after CS 121, cont.

- CS 228: Computational Learning Theory
- CS 229: Topics in the Theory of Computation
- AM 106: Applied Algebra & Combinatorics (every Fall)
- AM 107: Graph Theory & Combinatorics (every Spring)
- Logic: Math 141 (Mathematical Logic),
  144 (Model Theory),
  EMR 17 (deductive Logic),
  Phil 144 (Logic and Philosophy)
- Many courses in CS & Math at MIT.

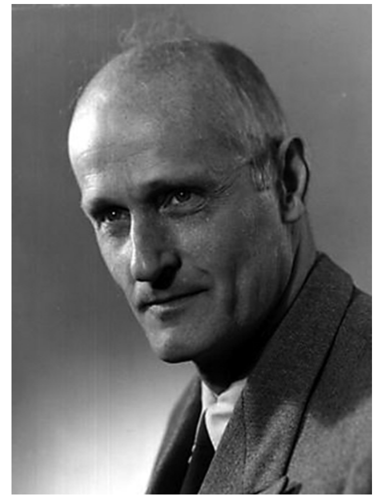
Theory Research Group

- Theory of Computation research group
  - Group webpage: `http://toc.seas.harvard.edu/`
  - Weekly seminar: Wednesdays at noon, usually in MD319, with pizza!
  - `http://toc.seas.harvard.edu/seminar.html`
- Also MIT Theory of Computation group and its seminars
  - `http://theory.csail.mit.edu/`
- Many research opportunities

Connections to the Rest of CS (Partial List)

Circuit Design (CS 141)	Finite Automata
Parsing + Compiling (CS 153)	Context-free Languages Pushdown Automata
Programming Languages (CS 152)	Regular Expressions Formalization in General
Natural Language + Linguistics (CS 187)	Grammars Finite Automata
Program Analysis + Synthesis (CS 153)	Uncomputability
Artificial Intelligence (CS 181,182)	Formal Systems, Logic

Remember the People



Library of Congress