Computational Complexity

Lecture 0

Introduction

Alexandra Kolla

Welcome to CSCI-7000-005

Administrative stuff

- Prerequisites: CSCI 2824 or equivalent,CSCI 3104/5454 or equivalent, familiarity with the notion of algorithm, running time, reduction, turing machine, basic notions of discrete math and probability.
- Course Website: link through my page http://home.cs.colorado.edu/~alko5368/
- Recommended reading: Arora-Barak "Computational Complexity: A Modern Approach".
- Office hours: Wedensdays 4-5 pm .
- Homeworks: ~6 homework sets(70%), final/final project(30%).

Today

- What is Computational Complexity all about and why do we care?
- Some examples of problems Complexity is interested in.
- Theoretical Applications ☺
- Course plan.

What is Complexity Theory?

- It is to computer science what theoretical physics is to electronics.

- Problems to be solved, algorithms to solve them: how much resources do they need? (time, storage space, randomness)

Complexity Theory

Classify problems according to the **computational resources** required

- running time
- storage space
- parallelism
- randomness
- rounds of interaction, communication, others…

Attempt to answer: what is computationally feasible with limited resources?

Complexity Theory

- Contrast with decidability: What is computable?
  - answer: some things are not
- We care about resources!
  - leads to many more subtle questions
  - fundamental open problems

The central questions

- Is finding a solution as easy as recognizing one?

  P = NP?

- Is *every* efficient sequential algorithm parallelizable?

  P = NC?

- Can *every* efficient algorithm be converted into one that uses a tiny amount of memory?

  P = L?

- Are there small Boolean circuits for *all* problems that *require* exponential running time?

  EXP in P/poly?

- Can *every* efficient randomized algorithm be converted into a deterministic algorithm one?

  P = BPP?

Central Questions

We *think* we know the answers to all of these questions …

… **but** no one has been able to prove that even a small part of this "world-view" is correct.

If we're wrong on any one of these then computer science will change dramatically

An (incomplete) overview

- Computational Complexity studies
  - Impossibility results (lower bounds). Eventually would like to prove the major conjectured lower bound P≠ NP, which would imply that thousands of natural combinatorial problems don't admit efficient algorithms.
  - Relations between the power of different computational resources (time, memory, randomness, communication) and the difficulties of different modes of computation (exact vs. approximate, worst case vs. average case…). Eg. would like to prove the conjecture P=BPP.

An (incomplete) overview

- Ultimately, we would like complexity theory to not only answer asymptotic worst-case questions (like P vs. NP) but also address the average and worst-case complexity of finite-sized instances, e.g.
  - The smallest boolean circuit that solves 3SAT on formulas with 300 variables has size more than 2^{70}
  - The smallest boolean circuit that can factor more than half of the 2000-digit integers, has size more than 2^{80}

If all of that happens…

- Develop unconditionally secure cryptosystems
- Understand what makes certain instances harder than others, develop more efficient algorithms
- Provide the mathematical language to talk about not only computations performed by computers, but also the behavior of discrete systems that evolve according to well-defined laws. (working of the cell, the brain, natural evolution, economic systems…)

Till then…

- Complexity theorists have had some success in proving lower bounds for restricted models of computation
- Some of the most interesting results in complexity theory regard connections between seemingly unrelated questions, yielding "unification" of the field
- We will see some of that next

Connections and Unifications

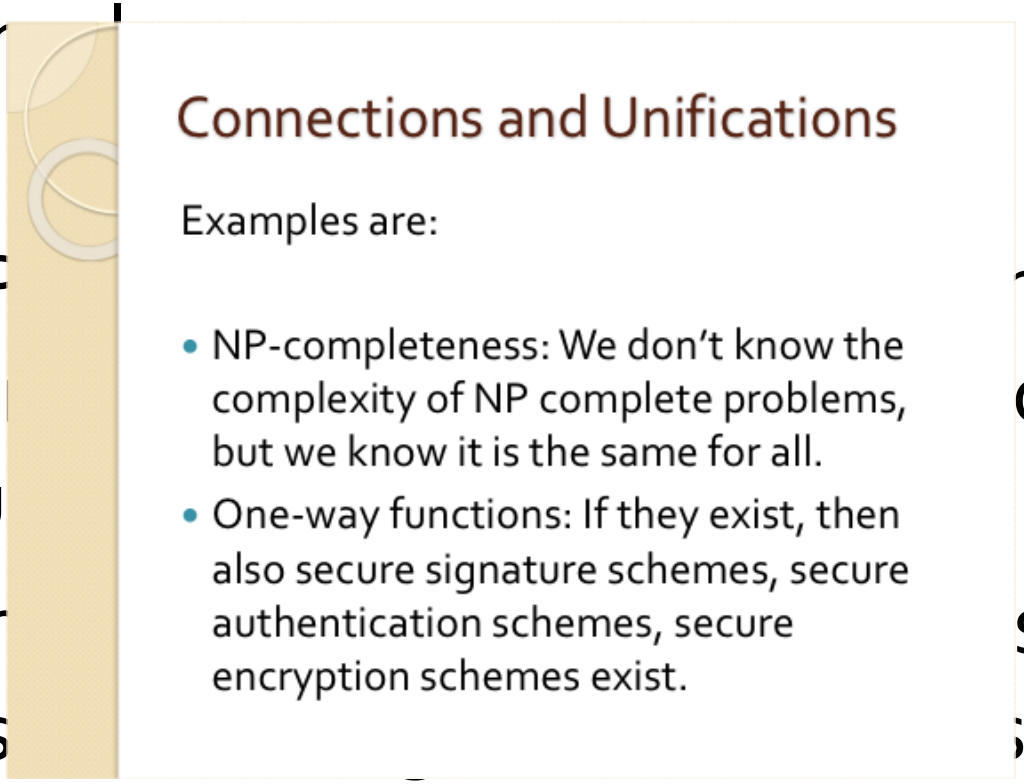
- Unconditional lower bounds have only been proven against restricted classes of algorithms or problems of very high complexity
- Most work in complexity theory is about connections between questions

Connections and Unifications

Exar

- NP
  co
  bu
- Or
  als

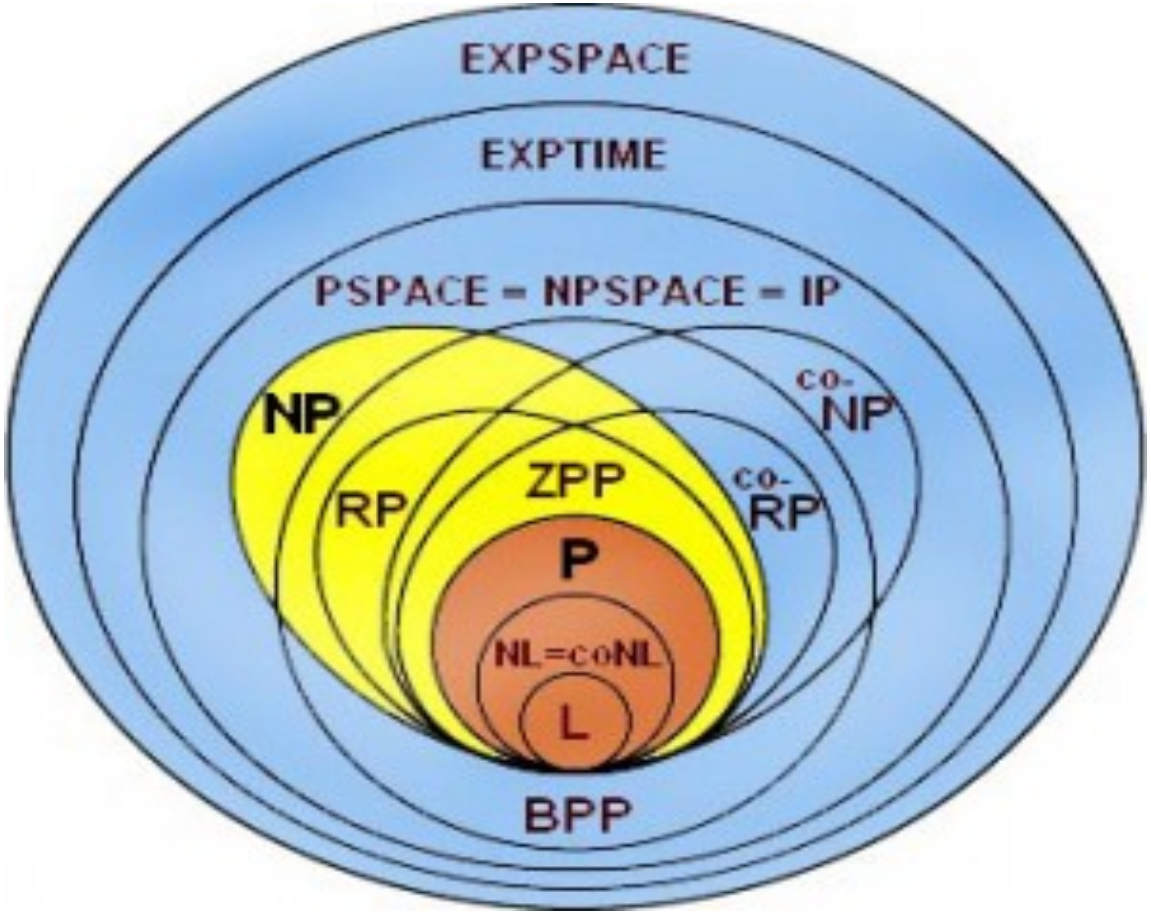
now the
oblems,
all.
st, then
, secure
authentication schemes, secure encryption schemes exist.



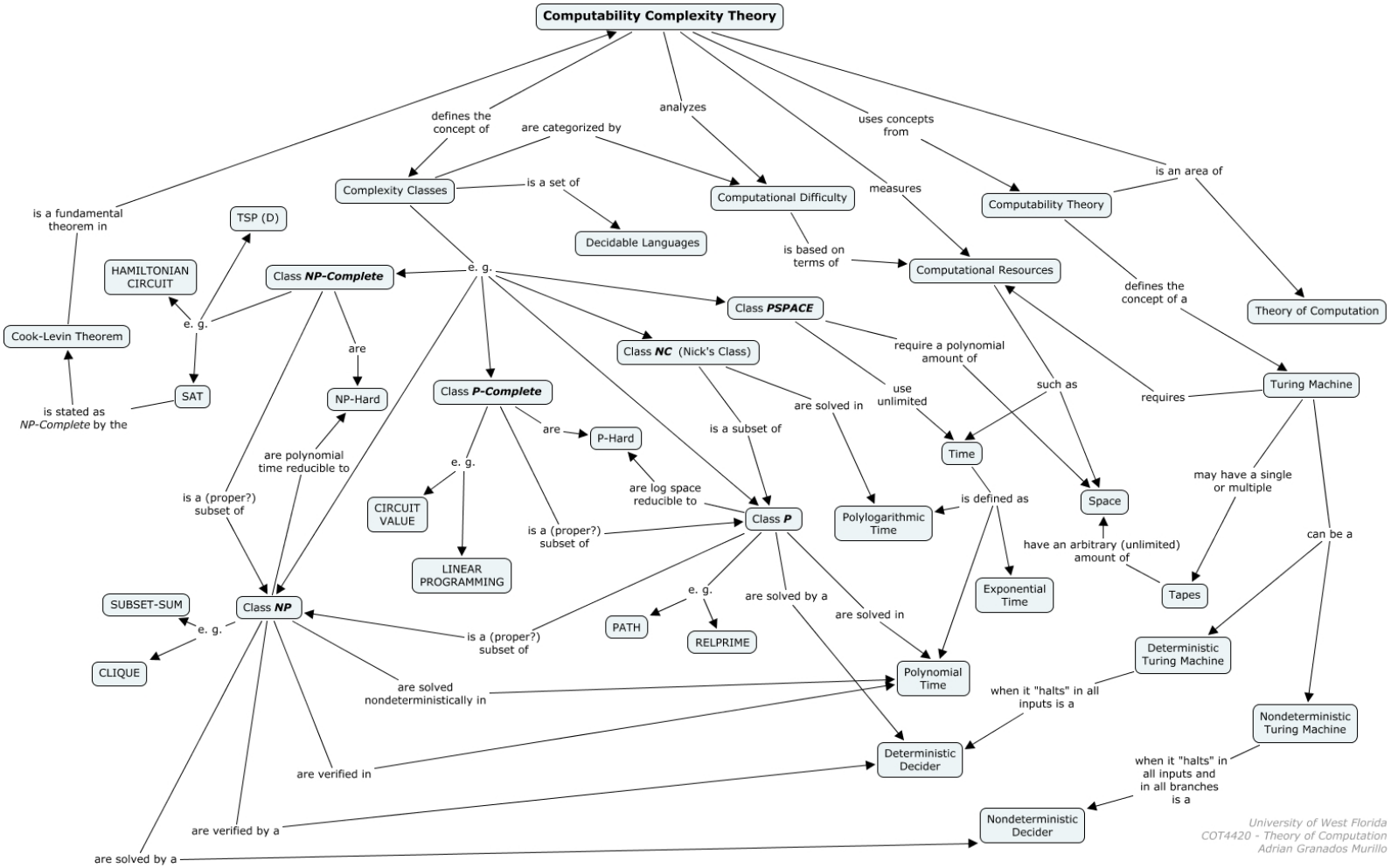
Connections and Unifications

- Probabilistically checkable proofs: characterization of NP that helps prove hardness of approximation.
- Derandomization: ideally would like to show that every randomized algorithm can be simulated deterministically. Can turn hardness assumption into algorithm.
- Worst case vs. average case: for certain problems can turn worst-case hardness into seemingly stronger (but in fact equivalent) average-case hardness.

Complexity Classes



Complexity Zoo!



Course plan

- The basics: look at the models in complexity theory, consider deterministic, non-deterministic, randomized, non-uniform and memory-bounded algorithms and the known relations between them. (about 4 weeks)
- Interactive proofs, IP=PSCPACE. (about 2 weeks)
- Expanders, Reingold's algorithm. (about 2 weeks).
- Derandomization, Pseudorandomness and Average-Case Complexity. (about 2 weeks)
- Unique Games Conjecture, PCP theorem… (tentative).
- Quantum Complexity Theory. (1-2 weeks)