Computational Complexity

Mohammad Mahmoody

7th Session
4 Feb 2014

Today Part 1:

- Recap important notions/theorems we saw so far

Formal Model for Computation

- Defined Turing Machines as formal model of computation

- Turing machine is “equivalent” to RAM model which is the basis for current computing machines.

- Turing machine (and RAM machines) can simulate each other and themselves with only a “polynomial-time” overhead.

- This gives rise to “Universal Turing Machine” that takes as input another Turing machine and runs it for a desired amount of time.

Complexity Classes and Languages

- Defined class $\mathbf{P}$: set of "easy" problems (languages) with YES/NO answer.

Solvable in polynomial time over input length

- Defined : set of problems with YES/NO answer where YES can always be accompanied with a "short" proof.

polynomial length over input length

- Note: if the answer is NO, there might not be a short proof for that…

- Easy to see that $\mathbf{P} \subseteq \mathbf{NP}$ but is it $\mathbf{P} = \mathbf{NP}$ or not? Big open question.

Reductions and **NP** Hardness

- Defined reductions as **one way** to relate hardness of two problems:
  1. Karp Reductions:
     - Only work for two **decision** problems L, L'
     - Given x, map it to f(x) and see if $f(x) \in L'$ or not

     $L \leq_p L'$
  2. Turing Reductions:
     - Work for essentially any problem:
     - Given a black-box that solves L' we use it to solve L in polynomial time

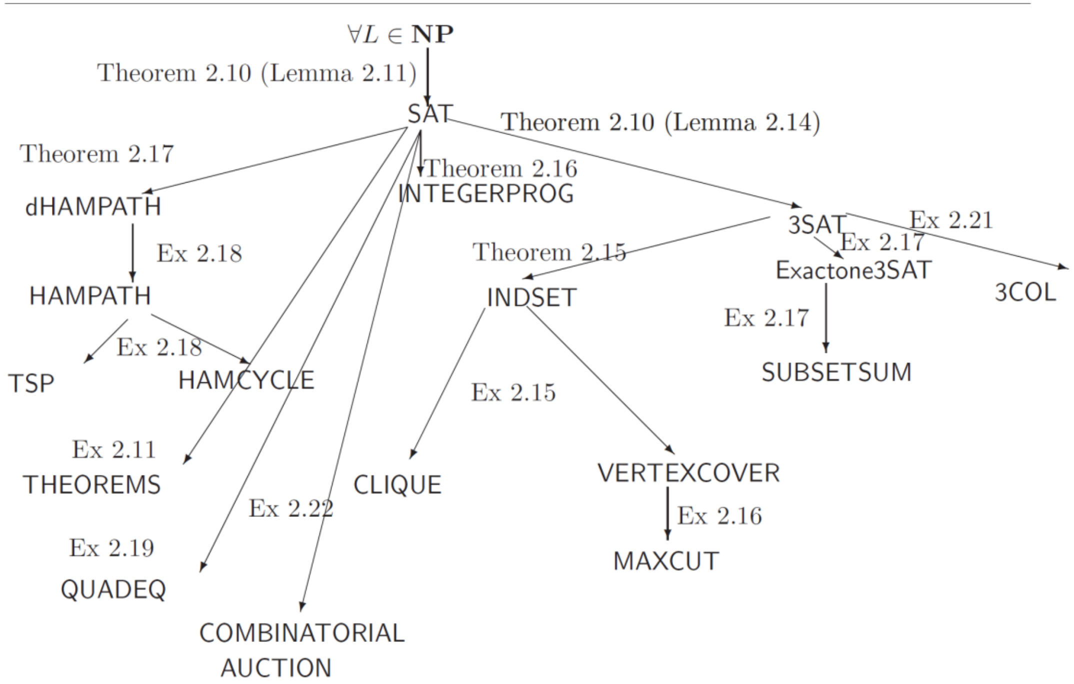
     $L \leq_T L'$
- For any notion of reduction, we can define **NP** hardness as:
  - L' is **NP** hard if: for all $L \in \mathbf{NP}$ there is a reduction from L to L'
- If an **NP** hard problem L' is in **NP** itself, then it is **NP** complete.
- **NP** complete are the "hardest" in **NP** so:
  - To answer $\mathbf{P} =_? \mathbf{NP}$ we can focus on **NP** complete problems only

$P \neq NP \implies \exists$ $L \notin P$, L is not NP-C, $L \in NP$

NP Completeness

- **NP** complete problems exist:
  - Easy to define "useless" **NP** complete problem TuringSAT
  - Cook-Levin Theorem: Circ-SAT is **NP** complete

- Web of reductions:
  Hundreds of problems are shown to be **NP** complete by showing how to reduce another **NP** complete problem to them.



Search vs. Decision

- For many problems the answer is not just YES/NO
- Example: For $L \in \mathbf{NP}$ and a given x we want to know whether $x \in L$ or not, and if $x \in L$ we want to find a "witness"
- Defined search problems in general and showed a connection back to languages in $\mathbf{NP}$ (whenever a "solution" can be easily verified).
- Trivial: reduction from Decision to Search
- Nontrivial: for all $\mathbf{NP}$ complete problems there is a reduction from Search to Decision
  - First proved it for Circ-SAT
  - Then used **the proof of Cook-Levin theorem** to generalize it to all $\mathbf{NP}$ complete problems

Today

$L \overset{?}{\in} NP$

$\bar{L} \le 3SAT$

Complement of $L = \bar{L} \in NP$

$\bar{L} \in NP$

3CNF formula

3SAT

- What is the complexity class of the following problems?

$3SAT = \{x \mid \ldots$ ~~not~~ $\ldots\}$

under Turing Red

- $L = \{x \mid x$ is a 3CNF formula that is not satisfiable$\}$

L is NP-hard

reduction take x ask x from B and flip answers of B

$\bar{L} \in NP$

- $L = \{x \mid x$ is a 3CNF formula that is a tautology (always satisfied)$\}$

$x \in L$ if $\exists$ .... made ...

$\forall$ .... and ...

$X \le Y \Rightarrow coX \le coY$

$P \subseteq NP$

$P \subseteq coNP$

Complement of a Language

- for any class $\mathbf{X}$, define $\mathbf{coX} = \{L \mid \overline{L} \in \mathbf{X}\}$

$coP \subseteq coNP$
$= P$

- If $L \in \mathbf{P} \Rightarrow \overline{L} \in \mathbf{P}$ as well so: $\mathbf{P} = \mathbf{coP}$
- If $L \in \mathbf{NP} \Rightarrow \overline{L} \in_? \mathbf{NP}$?

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

- $\mathbf{coNP} = \{L \mid \overline{L} \in \mathbf{NP}\}$
  coNP is not the complement of **NP**
  NP and **coNP** both include **P**

- Can define **coNP** hardness and **coNP** completeness
  - $\overline{\text{CircSAT}}$ is **coNP** complete

coNP hard L:
if all $S \in coNP$
reduce to L

coNP complete L:
co-NP hard +
$L \in co\text{-}NP$

$\mathbf{NP}$: Nondeterministic Polynomial Time

NDTM has *two* transition functions δ_0 and δ_1, and a special state denoted by q_{accept}. When an NDTM M computes a function, we envision that at each computational step M makes an arbitrary choice as to which of its two transition functions to apply. For every input x, we say that $M(x) = 1$ if there *exists* some sequence of these choices (which we call the *non-deterministic choices* of M) that would make M reach q_{accept} on input x.

Definition 2.5 For every function $T : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$, we say that $L \in \mathbf{NTIME}(T(n))$ if there is a constant $c > 0$ and a $c \cdot T(n)$-time NDTM M such that for every $x \in \{0,1\}^*$, $x \in L \Leftrightarrow M(x) = 1$ $\diamondsuit$

Theorem 2.6 $\mathbf{NP} = \cup_{c \in \mathbb{N}} \mathbf{NTIME}(n^c)$