Computational Complexity

Mohammad Mahmoody

5th Session

28 Jan 2014

1st Assignment

- Due tomorrow 5pm

- After that, can email me (mahmoody@gmail.com) a typed or scanned version by Monday 5pm
  (by Friday 5pm 80%, Sat 5pm 60%, Sun 5pm 40%, Mon 5pm 20%)

- Monday 5pm: Abbas (TA) will solve the problems
  or sketch of answers will be uploaded

Last Time

- $\mathbf{NP}$-complete problems exist
  (TM-SAT: artificial language tailored to the definition of $\mathbf{NP}$)

- Natural $\mathbf{NP}$-complete problems exist: Cook-Levin theorem
  (Circuit-SAT is $\mathbf{NP}$-complete)

Today

- $\mathbf{NP}$-completeness is everywhere (Karp's famous paper)
- Search Problems vs. Decision Problems
- Complement of Languages

Recall Cook-Levin Theorem

- Theorem: Circ-SAT is **NP**-complete given Circuit C: is there x: C(x)=1

witness

- Assignment: 3SAT is **NP**-complete

x, y, z, t

$(\neg x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z) \wedge (t \vee x \vee \neg y)$

clause

- Karp 1972: 21 natural combinatorial problems are all **NP**-complete !

literals

Gary Johnson

More **NP** complete problems

- Recall following problems in **NP**
  - IND-SET
  - CLIQUE
  - Vertex-COVER

input (G, k) : is there an independent set of size $\geq k$ in G.

graph

- We showed that:
  - IND-SET $\leq_p$ CLIQUE and CLIQUE $\leq_p$ IND-SET
  - IND-SET $\leq_p$ V-Cover and V-Cover $\leq_p$ IND-SET

- So if we prove any of these problems (e.g. X) to be **NP** complete then other ones (e.g. Y) will be **NP** complete as well, because for all $L \in NP$:
  $L \leq_p X \leq_p Y$ and so $L \leq_p Y$

IND-SET is **NP** complete

Use 3-SAT

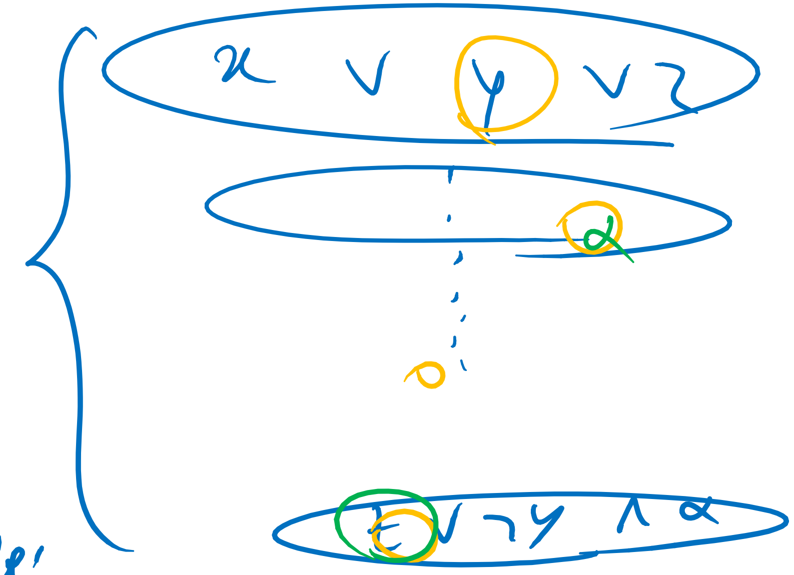
Assume A is Algorithm for IND-SET   goal: Solving 3-SAT

Want function f: takes (3-SAT) question: $X \xrightarrow{f} f(x)$

$f(x) \in$ IND-SET $\iff X \in$ 3-SAT

k clauses

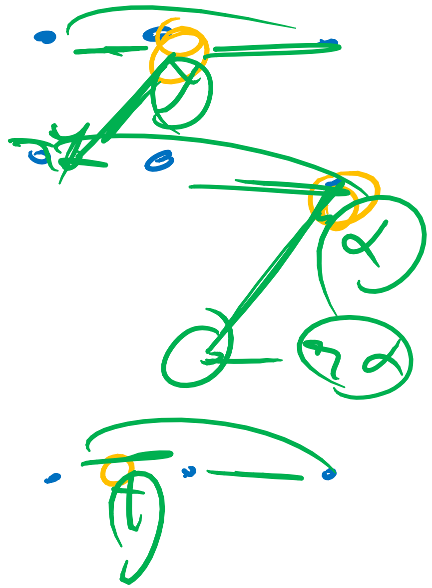
m variables



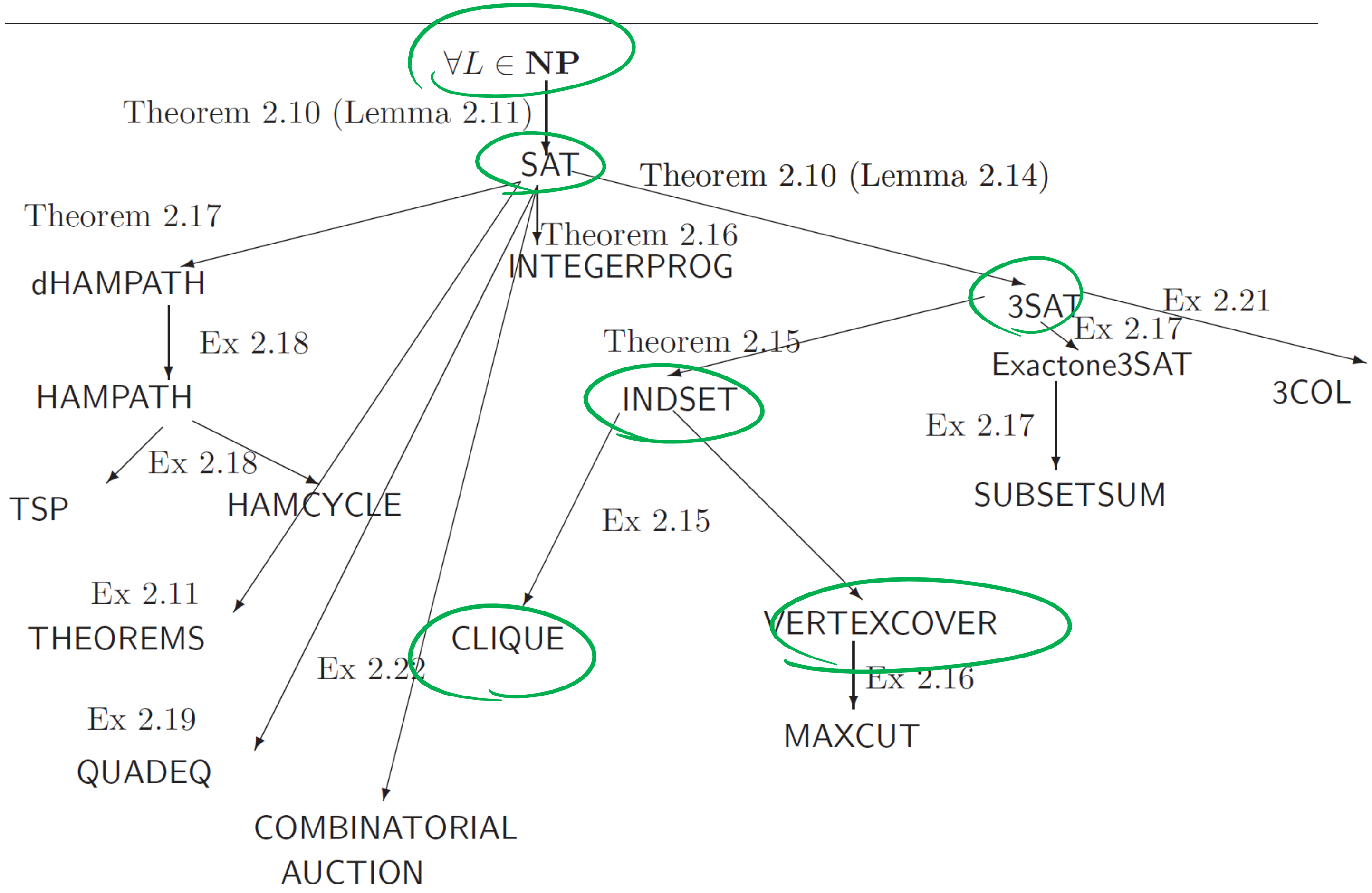
f

Graph G

nodes: 3k



Web of Reductions



Search vs. Decision

- So far we defined problems as “decision problems”
- What about: given circuit C, finding w such that $C(w) = 1$ (not just knowing whether ~~x~~ w exists or not)
- Let **Search-Circ-SAT** be search version of **Circ-SAT**
- Easy: If we can solve **Search-Circ-SAT** in polynomial time $\Rightarrow$ can solve (Decision) **Circ-SAT** in polynomial time
- The reduction is very similar to Karp-Reduction, **<u>but</u>** note that **Search-Circ-SAT** is not a decision problem.
- Question: What if we can solve **Circ-SAT** $\in \mathbf{P}$?

Search vs Decision for **NP**

- Any relation R defines a search problem:
  For every input x we want find y such that $(x, y) \in R$

polynomial-time algo V

- Most natural search problems :
  1. Given y it is easy to verify that $(x, y) \in R$
  2. Thus, if we define $L_R = \{x \mid \exists\, y\colon (x, y) \in R\}$ then $L_R \in \mathbf{NP}$.

$\exists$ verification V s.t. $x \in L$ iff $\exists$ w (x, w) accepted by V

- Conversely: For every $L \in \mathbf{NP}$ there is a natural search problem
  $R_L = \{(x, w) \mid w \text{ is a witness for } x\}$

Search Vs. Decision: **Circ-SAT**

- Theorem: If we could solve (Decision) **Circ-SAT** in polynomial time we can also solve the search version: **Search-Circ-SAT** in polynomial time

- **Proof:** Lets call B Algorithm for Circuits SAT.

run B ≤ n times.

input circuit

output

① ask C from B — No / Yes?

② try x = true : $C_{x=true}$

ask $C_{x=true}$ from B — Yes / No.

if yes. fix x = true

No fix x = false

$\Longrightarrow C' = C_{x=true/false}$