EECS 336: Introduction to Algorithms	**Lecture 12**
P vs. NP (cont.)	NP, CIRCUIT-SAT, 3-SAT

Reading: Chapter 8; guide to reductions

Last time:

- 3-SAT $\leq_{\mathcal{P}}$ INDEP-SET
- 3-SAT $\leq_{\mathcal{P}}$ HC

Today:

- NP $\leq_{\mathcal{P}}$ CIRCUIT-SAT $\leq_{\mathcal{P}}$ LE3-SAT $\leq_{\mathcal{P}}$ 3-SAT.

Notorious Problem: NP

input:

- decision problem verifier program VP.
- polynomial $p(\cdot)$.
- decision problem instance: x

output:

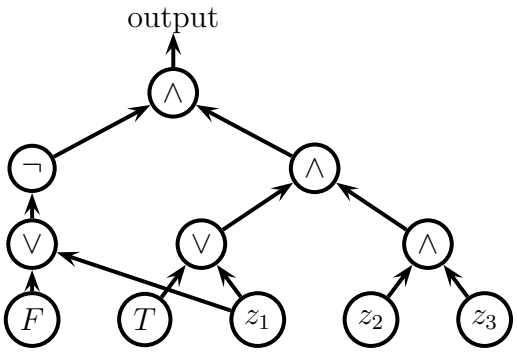
- "Yes" if exists certificate c such that $VP(x, c)$ has "verified = true" at computational step $p(|x|)$.
- "No" otherwise.

Fact: NP is $\mathcal{NP}$-complete.

Agenda: Find a first simple $\mathcal{NP}$-complete problem.

Circuit Satisfiability

Example:



Problem 4: CIRCUIT-SAT

input: boolean circuit $Q(\mathbf{z})$

- directed acyclic graph $G = (V, E)$
- internal nodes labeled by logical gates:

  "and", "or", or "not"

- leaves labeled by variables or constants

  $T, F, z_1, \ldots, z_n$.

- root r is output of circuit

output:

- "Yes" if exists $\mathbf{z}$ with $Q(\mathbf{z}) = T$
- "No" otherwise.

Lemma: CIRCUIT-SAT is $\mathcal{NP}$-hard.

Part I: forward instance construction

convert NP instance (VP, p, x) to CIRCUIT-SAT instance Q

- $VP(\cdot, \cdot)$ polynomial time

  $\Rightarrow$ computer can run it in poly steps.

- each step of computer is circuit.
- output of one step is input to next step
- unroll $p(|x|)$ steps of computation

  $\Rightarrow$ $\exists$ poly-size circuit $Q'(\mathbf{x}, \mathbf{c}) = VP(x, c)$

- hardcode $\mathbf{x}$: $Q(\mathbf{c}) = Q'(\mathbf{x}, \mathbf{c})$

Part II-III: backward/forward certificate construction

- $\mathbf{z} = \mathbf{c}$

PICTURE

LE3-SAT

"CIRCUIT-SAT $\leq_{\mathcal{P}}$ LE3-SAT $\leq_{\mathcal{P}}$ 3-SAT"

Problem 5: LE3-SAT

"like 3-SAT but at most 3 literals per or-clause"

Note: $\leq_{\mathcal{P}}$ is transitive: if $Y \leq_{\mathcal{P}} X$ and $X \leq_{\mathcal{P}} Z$ then $Y \leq_{\mathcal{P}} Z$.

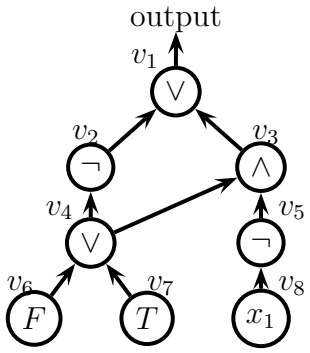
Recall: NP $\leq_{\mathcal{P}}$ CIRCUIT-SAT

Plan: CIRCUIT-SAT $\leq_{\mathcal{P}}$ LE3-SAT $\leq_{\mathcal{P}}$ 3-SAT

conclusion: 3-SAT is $\mathcal{NP}$-hard

Lemma: CIRCUIT-SAT $\leq_{\mathcal{P}}$ LE3-SAT

Example:



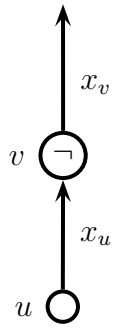
Proof: (reduce from CIRCUIT-SAT)

Part I: forward instance construction

convert CIRCUIT-SAT instance Q into 3-SAT instance f

should be f should be satisfiable iff Q has input that makes it's output true

- variables x_v for each vertex of Q.
- encode gates
  - **not**: if v not gate with input from u



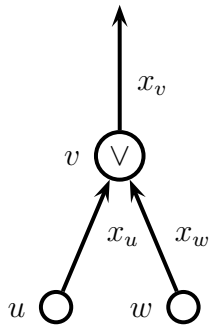
need $x_v = \bar{x}_u$

$x_v \setminus x_u$	0	1
0	0	1
1	1	0

$\Rightarrow$ add clauses $(x_v \vee x_u) \wedge (\bar{x}_v \vee \bar{x}_u)$

  - **or**: if v is or gate from u to w

need $x_v = x_u \wedge x_w$



$x_v \setminus x_u x_w$	00	01	11	10
0	1	0	0	0
1	0	1	1	1

$\Rightarrow$ add clauses $(\bar{x}_v \vee x_u \vee x_w) \wedge (x_v \vee \bar{x}_u) \wedge (x_v \vee \bar{x}_w)$

  - **and:** if v is and gate from u to w

$\Rightarrow$ add clauses $(x_v \vee \bar{x}_u \bar{x}_w) \wedge (\bar{x}_v \vee x_u) \wedge (\bar{x}_v \vee x_w)$.

- **0:** if v is 0 leaf.

  need $x_v = 0$

  $\Rightarrow$ add clause $(\bar{x}_v)$

  need $x_v = 1$

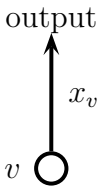
- **1:** if v is 1 leaf.

  $\Rightarrow$ add clause (x_v)

- **literal:** if v is literal z_j

  $\Rightarrow$ do nothing
  [[x_v *can be anything*]]

- **root:** if v is root



need $x_v = 1$

$\Rightarrow$ add clause (x_v).

Runtime Analysis: construction is polynomial time.

- at most 3 clauses in f per node in Q.

Part II: backward certificate construction

convert LE3-SAT assignment $\mathbf{x}$ to CIRCUIT-SAT assignment $\mathbf{z}$

1. read $\mathbf{z}$ from $\mathbf{x}$ corresponding to literals.

Claim: $f(\mathbf{x}) \Rightarrow Q(\mathbf{z})$

- f constrains variables x_i to "proper circuit outcomes" and root is True.

$\Rightarrow$ $Q(\mathbf{z})$ is True.

Part III: forward certificate construction

convert CIRCUIT-SAT assignment $\mathbf{z}$ to LE3-SAT assignment $\mathbf{x}$

1. simulate Q on $\mathbf{z}$
2. read $\mathbf{x}$ from values of gates in circuit.

Claim: $Q(\mathbf{z}) \Rightarrow f(\mathbf{x})$

- by construction, $f(\cdot)$ encodes proper working ciruit that evaluates to True.
- Since $Q(\mathbf{z})$ is true, and $\mathbf{x}$ is from simulation of $Q(\cdot)$, $f(\mathbf{x})$ is true.

QED

Lemma: LE3-SAT $\leq_{\mathcal{P}}$ 3-SAT

> Note: 3-SAT $\leq_{\mathcal{P}}$ LE3-SAT is obvious, LE3-SAT $\leq_{\mathcal{P}}$ 3-SAT is not.

Part I: forward instance construction convert LE3-SAT instance f into 3-SAT instance f'

- $f' \leftarrow f$ rename variables to
- add variables w_1, w_2 [[*idea:* $w_i = F$]]
- add w_i to 1- and 2-clauses

  $(l_1) \Rightarrow (l_1 \vee w_1 \vee w_2)$.

  $(l_1 \vee l_2) \Rightarrow (l_1 \vee l_2 \vee w_1)$.
- ensure $w_i = 0$ add variables y_1, y_1 and clauses:

  $(\bar{w}_i \vee y_1 \vee y_2)$

  $(\bar{w}_i \vee \bar{y}_1 \vee y_2)$

  $(\bar{w}_i \vee y_1 \vee \bar{y}_2)$

  $(\bar{w}_i \vee \bar{y}_1 \vee \bar{y}_2)$
- denote $\mathbf{x}' = (\mathbf{x}, w_1, w_2, y_1, y_2)$

Runtime Analysis: construction is polynomial time.

Part II: backward certificate construction

$\mathbf{x}' \Rightarrow \mathbf{x}$

1. read $\mathbf{x}$ from $\mathbf{x}'$ (all but last 4 variables).

Claim: $f'(\mathbf{x}') \rightarrow f(\mathbf{x})$

- Let $\mathbf{x}' = (\bar{z}, w_1, w_2, y_1, y_2)$.
- $f'(\mathbf{x}') = \text{True}$

  $\Rightarrow$ by construction, $w_i = \text{False}$

  $\Rightarrow f'(\mathbf{x}, F, F, y_1, y_2) \overset{\text{simplify}}{\Longrightarrow} f(\mathbf{x})$

  $\Rightarrow f(\mathbf{x}) = \text{True}$.

Part III: forward certificate construction

$\mathbf{x} \Rightarrow \mathbf{x}'$

1. set $\mathbf{x}' = (\mathbf{x}, F, F, F, F)$

Claim: $f(\mathbf{x}) \rightarrow f'(\mathbf{x}')$

- $f(\mathbf{x}) = \text{True}$
  - $f(\mathbf{x}, w_1, w_2, y_1, y_2) \overset{\text{simplify}}{\Longrightarrow}$ "clauses with only w_i and y_i"
  - with $w_i = F$ and $y_i = F$ (or anything) these are true. **QED**