Reading: 8.1-8.4

Last time:

- 3-SAT $\leq_p$ INDEPENDENT-SET

Today:

- NP $\leq_p$ CIRCUIT-SAT $\leq_p$ 3-SAT

Notorious Problem: NP

input:

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

output:

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

Problem 4: 3-SAT

input: boolean formula $f(z)$

- in conjunctive normal form (CNF)

output:

- “Yes” if assignment $z$ with $f(z) = T$ exists
- “No” otherwise.

Note: 2 steps to NP-completeness

1. $X \in \mathcal{NP}$
2. $X$ is $\mathcal{NP}$-hard (via reduction)

3 steps to reduction

1. construction
2. runtime of construction
3. correctness of construction (iff)

Note: algorithms in reductions:

<table>
<thead>
<tr>
<th>3-SAT</th>
<th>INDEPENDENT-SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>input: $f$ $\Rightarrow$ $G,D$</td>
<td>output: $z$ $\iff$ $S$</td>
</tr>
</tbody>
</table>
Circuit Satisfiability

Example:

️

Problem 4: CIRCUIT-SAT

input: boolean circuit $Q(z)$

- 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 $z$ with $Q(z) = T$
- “No” otherwise.

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

Proof: (reduce from NP)

- goal: convert NP instance $(V_P, p, x)$ to CIRCUIT-SAT instance $Q$
- $V_P(\cdot, \cdot)$ polynomial time

⇒ 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

⇒ $\exists$ poly-size circuit $Q'(x, c) = V_P(x, c)$

- hardcode $x$: $Q(c) = Q'(x, c)$
- Conclusion: $Q$ is sat iff exists $c$ with $V_P(x, c) = \text{“verified”}$.  

QED
LE3-SAT

“CIRCUIT-SAT $\leq_P$ LE3-SAT $\leq_P$ 3-SAT”

Problem 5: LE3-SAT

“like 3-SAT but at most 3 literals per or-clause”

Note: $\leq_P$ is transitive: if $Y \leq_P X$ and $X \leq_P Z$ then $Y \leq_P Z$.

Recall: NP $\leq_P$ CIRCUIT-SAT

Plan: CIRCUIT-SAT $\leq_P$ LE3-SAT $\leq_P$ 3-SAT

Lemma: CIRCUIT-SAT $\leq_P$ LE3-SAT

Example:

Proof: (reduce from CIRCUIT-SAT)

Step 1: convert CIRCUIT-SAT instance $Q$ into 3-SAT instance $f$

- variables $x_v$ for each vertex of $Q$.
- encode gates
  - not: if $v$ not gate with input from $u$
  - or: if $v$ is or gate from $u$ to $w$ need $x_v = x_u 
  - and: if $v$ is and gate from $u$ to $w$ need $x_v = x_u \land x_w$
  - 0: if $v$ is 0 leaf.

<table>
<thead>
<tr>
<th>$x_v \setminus x_u x_w$</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$\Rightarrow$ add clauses $(x_v \lor x_u \lor x_w) \land (x_v \lor \bar{x}_u) \land (x_v \lor \bar{x}_w)$
⇒ add clause \((\bar{x}_v)\)
need \(x_v = 1\)

- **1:** if \(v\) is 1 leaf.
⇒ add clause \((x_v)\)

- **literal:** if \(v\) is literal \(z_j\)
⇒ do nothing

- **root:** if \(v\) is root

generate:

\[
\text{output} \\
\begin{array}{c}
x_v \\
v
\end{array}
\]
need \(x_v = 1\)
⇒ add clause \((x_v)\).

**Step 2:** construction is polynomial time.

- at most 3 clauses in \(f\) per node in \(Q\).

**Step 3:** construction is correct (i.e., \(Q\) is sat iff \(f\) is sat.)

- \(f\) constrains variables \(v_i\) to “proper circuit outcomes”.
- if exists \(z\) s.t. \(f(z) = T\),
then can read \(x\) from \(z\) and \(z\) encodes proper circuit outcome to make \(Q\) output \(T\) for this \(x\).
- if \(Q\) outputs \(T\) for some \(x\)
then can map \(x\) and values at nodes to variables \(z\) such that \(f(z)\) is true.

**Lemma:** \(\text{LE3-SAT} \leq_p \text{3-SAT}\)

**Step 1:** convert LE3-SAT instance \(f'\) into 3-SAT instance \(f\)

- \(f \leftarrow f'\)
- add variables \(w_1, w_2\)
- add \(w_i\) to 1- and 2-clauses
\((l_1) \Rightarrow (l_1 \lor w_1 \lor w_2).\)
\((l_1 \lor l_2) \Rightarrow (l_1 \lor l_2 \lor w_1).\)
- ensure \(w_i = 0\) add variables \(y_1, y_1\) and clauses:
\((\bar{w}_i \lor y_1 \lor y_2)\)
\((\bar{w}_i \lor \bar{y}_1 \lor y_2)\)
\((\bar{w}_i \lor y_1 \lor \bar{y}_2)\)
\((\bar{w}_i \lor \bar{y}_1 \lor \bar{y}_2)\)

**Step 2:** construction is polynomial time.

**Step 3:** \(f\) is sat iff \(f'\) is sat.

- given satisfying assignment \((\bar{z}, w_1, w_2, y_1, y_2)\) to \(f\),
⇒ \(w_i = F\) by construction.
⇒ \(f(\bar{z}, F, F, y_1, y_2) \Rightarrow f(\bar{z})\)
⇒ \(f\) is sat.
- given satisfying assignment \(\bar{z}\) to \(f'\),
  - \(f(\bar{z}, w_1, w_2, y_1, y_2) \Rightarrow \text{“clauses with only } w_i \text{ and } y_i\)"
  - set \(w_i = F\) and \(y_i = F\) (or anything) to satisfy.

QED