Reading: Chapter 8; guide to reductions

Last time:

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

Today:

- CIRCUIT-SAT $\leq_{\mathcal{P}}$ LE3-SAT (cont)
- LE3-SAT $\leq_{\mathcal{P}}$ 3-SAT
- $\mathcal{NP}$ review.

Lemma 0.1 *CIRCUIT-SAT* $\leq_{\mathcal{P}}$ *LE3-SAT*

Part I:: *forward instance construction*

$Q \Rightarrow f$

"f encodes proper working gates and output $Q(\mathbf{z}) = $ *true"*

LE3-SAT

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

Problem 5: LE3-SAT

"like 3-SAT but <u>at most</u> 3 literals per or-clause"

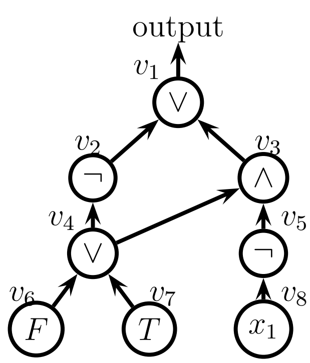
Note: $\leq_{\mathcal{P}}$ is transitive: if $Y \leq_{\mathcal{P}} X$ and $X \leq_{\mathcal{P}} Z$ then $Y \leq_{\mathcal{P}} Z$.

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

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

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

Example:

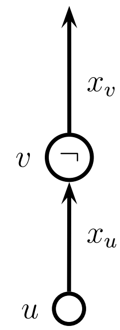


Proof: (reduce from CIRCUIT-SAT)

Part I: forward instance construction

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

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



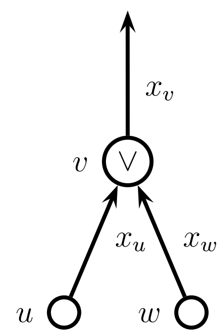
need $x_v = \bar{x}_u$

$x_v \setminus x_u$	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 $x_v = 0$

⇒ add clause $(\bar{x}_v)$

need $x_v = 1$

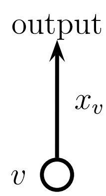
- **1:** if v is 1 leaf.

  ⇒ add clause (x_v)

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

  ⇒ do nothing

- **root:** if v is root



need $x_v = 1$

⇒ 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.

⇒ $Q(\mathbf{z})$ 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.

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

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
- 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**

$\mathcal{NP}$ hardness

"proof by contradition: solve hard problem Y with blackbox for X, so X must be hard"

One-call Reductions

1. forward instance construction:
   $y \Rightarrow x^y$
2. backward certificate construction:
   x^y is yes $\Rightarrow$ y is yes.
3. forward certificate construction:
   y is yes $\Rightarrow$ x^y is yes

Conclusion: y is yes if and only if x^y is yes.

Compare:

- show
  (a) x^y is yes $\Rightarrow$ y is yes
  (b) x^y is no $\Rightarrow$ y is no.
- show
  (a) x^y is yes $\Rightarrow$ y is yes
  (b) y is yes $\Rightarrow$ x^y is yes.

Deciding is as hard as optimizing

Proof: (reduction via binary search)

- given
  - instance x of X
  - black-box $\mathcal{A}$ to solve X_d
- search(A, B) = find optimal value in $[A, B]$.
  - $D = (A + B)/2$
  - run $\mathcal{A}(x, D)$
  - if "yes", search(A, D)
  - if "no", search(D, B)

Finding solution is as hard as deciding

Example: 3-SAT

1. if f is satisfiable $\exists \mathbf{z}$ s.t. $f(\mathbf{z}) = T$
2. guess $z_n = T$
3. let $f'(z_1, .., z_{n-1}) = f(z_1, .., z_{n-1}, T)$
4. simplify f' and convert from LE3-SAT to 3-SAT $\Rightarrow g$
5. if g is satisfiable, repeat (2) on f'
6. if f' is unsatisfiable,
   repeat (2) on $f''(z_1, \ldots, z_{n-1}) = f(z_1, \ldots, z_{n-1}, F)$ simplified.

Example: INDEP-SET