EECS 336: Lecture 12: Introduction to Algorithms

Deriving NP: NP, CIRCUIT-SAT

Reading: 8.3

Last Time:

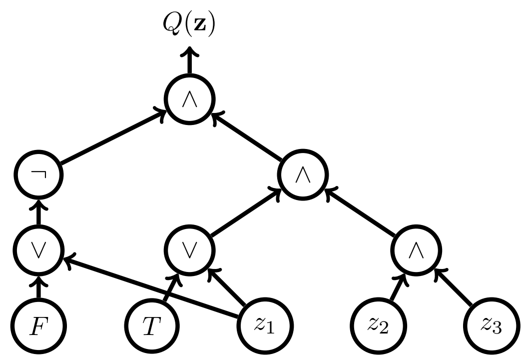
- decision problems
- $\mathcal{NP}$ problems
- "Notorious Problem" NP
- NP $\leq_P$ CIRCUIT-SAT

Today:

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

Circuit Satisfiability

Example:



Problem: 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

  $T, F, z_1, ..., z_n$.
- root r is output of circuit

output:

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

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

Part I: forward instance construction

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

- $\text{VP}(\cdot, \cdot)$ polynomial time

  $\Rightarrow$ computer can run it in poly steps.
- each step of computer is circuit.
- output of one step is input of next step

- unroll $p(|x|)$ steps of computation

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

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

Part II-III: backward/forward certificate construction

- $\mathbf{c} \Leftrightarrow c$

LE3-SAT

Problem: LE3-SAT

"like 3-SAT but **at most** 3 literal per or-clause"

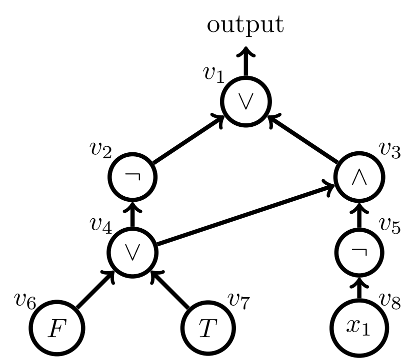
Note: $\leq_P$ is transitive: if $Y \leq_P X$ and $X \leq_P Z$ then $Y \leq_P Z$.

Recall: NP $\leq_P$ CIRCUIT-SAT

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

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

Example:

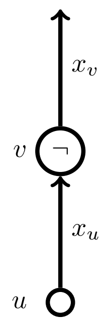


Proof: (reduce from CIRCUIT-SAT)

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



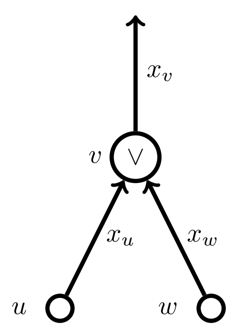
need $x_v = \bar{x}_u$

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

$\implies$ 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

$\implies$ 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 and gate from u to w

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

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

  need $x_v = 0$

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

  need $x_v = 1$

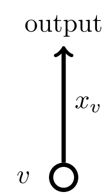
- **1:** if v is 1 leaf.

  $\implies$ add clause (x_v)

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

  $\implies$ do nothing

- **root:** if v is root



need $x_v = 1$

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

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

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

Part III: forward certificate construction convert CIRCUIT-SAT assignment z to LE3-SAT assignment $\mathbf{x}$

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

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

- 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

Theorem: LE3-SAT $\leq_P$ 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

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

$$(l_1 \vee l_2) \implies (l_1 \vee l_2 \vee w_1).$$

- ensure $w_i = 0$ add variables y_1, y_2 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 $\mathbf{x}' = (\mathbf{x}, w_1, w_2, w_3, y_1, y_2)$

Runtime Analysis: construction is polynomial time.

Part II: backward certificate construction

$\mathbf{x}' \implies \mathbf{x}$

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

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

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

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

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

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

Part III: forward certificate construction

$\mathbf{x} \implies \mathbf{x}'$

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