Alternate Definition of NP

Polynomial-time verification algorithm for decision problem X.

For all x, X(x) = YES $\longleftrightarrow$ there is a y such that X-VERIFY(x,y) = YES

→ polynomial in size of x

certificate (evidence)

HC-VERIFY(G, p)

poly check that each v in G appears at least once in p

poly check that each v in G appears at most once in p

poly for each $(v_i, v_j) \in p$, check that edge (v_i, v_j) exists in G and edge from last to first exists in G

if all YES, return YES

else return NO

length = n

if G has Hamiltonian cycle c, then HC-VERIFY(G, c) = YES

if G has no Hamiltonian cycle, then HC-VERIFY(G, p) = NO for all p

NP = set of problems X s.t. there is a polynomial-time verification algorithm for X

NP′ = set of problems X s.t. there is a non-deterministic polynomial-time solution for X

N P

allow coin flips

X(x) = YES

↕

P(X-RANDOM(x) = YES) > 0

HC ∈ NP′ : HC-RANDOM(G)

→ randomly permute vertices to get $v_0, \ldots, v_{n-1}, v_0$

for i = 0 to n−1

if no edge $(v_i, v_{(i+1)\%n})$ output NO

output YES

if G has HC c, HC-RANDOM picks c with prob $\frac{1}{n!} > 0$ and outputs YES

if G has no HC, no matter what permutation is chosen, always outputs NO

NP = NP′

$X(x) \overset{=Y}{\to} \exists y$ s.t. |y| is poly in |x| and X-VERIFY(x,y) = Y

$X(x) = N \to \forall y$, X-VERIFY(x,y) = N

NP ⊆ NP′ : Let X ∈ NP. Then ∃ poly-time verifier X-VERIFY(x,y)

so ∃ poly p(n) s.t. X-VERIFY runs in p(|x|+|y|) time and poly g(n) s.t. certificates are of length g(|x|)

Write non-deterministic alg for X: X-RANDOM(x)

→ randomly create y of length ≤ g(|x|)

output X-VERIFY(x, y)

finite # of ys of size g(|x|)

$X(x) = \text{YES} \to \exists y$ s.t. X-VERIFY$(x,y) = \text{YES} \to$ X-RANDOM picks y with prob > 0
$\to$ X-RANDOM outputs Y w/ prob > 0

$X(x) = \text{NO} \to \forall y$, X-VERIFY$(x,y) = \text{NO} \to$ X-RANDOM (x) outputs NO always
$\to$ X-RANDOM (x) outputs YES w/prob 0

NP′ ⊆ NP : Let $X \in NP'$. Then $\exists$ poly-time non-deterministic X-RANDOM(x).
↳ time $p(|x|)$ for some poly p

Write verifier X-VERIFY(x, y)
poly — simulate X-RANDOM(x) use y as random bits

$X(x) = \text{YES} \to P(\text{X-RANDOM}(x) = \text{YES}) > 0$
$\to$ some sequence of random bits y makes X-RANDOM(x) = YES — max size $p(|x|)$
$\to$ X-VERIFY(x,y) = YES

$X(x) = \text{NO} \to P(\text{X-RANDOM}(x) = \text{NO}) = 0$
$\to$ all y make X-VERIFY(x,y) = NO

CIRCUIT-SAT and SAT

SAT → given φ, determine if some truth assign to vars makes φ TRUE

CIRCUIT-SAT → given a combinational circuit, determine if some combination of inputs makes output T

SAT is NP-complete:

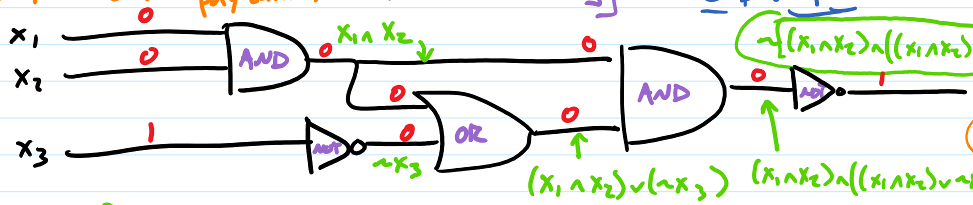
NP-complete: $X \in NP$ ✓
✓ $Y \le_p X$ for all $Y \in NP$

1) $SAT \in NP$ ✓
2) $\text{CIRCUIT-SAT} \le_p SAT$ ✓

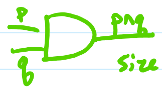
[Goal: given circuit C, create φ s.t. C is satisfiable $\Updownarrow$ φ is satisfiable]

$Y \le_p X$ means
$\exists$ some alg for Y using poly time + poly calls X

[$X \in NP$
Y is NP-complete ✓
$Z \le_p Y \le_p X$]

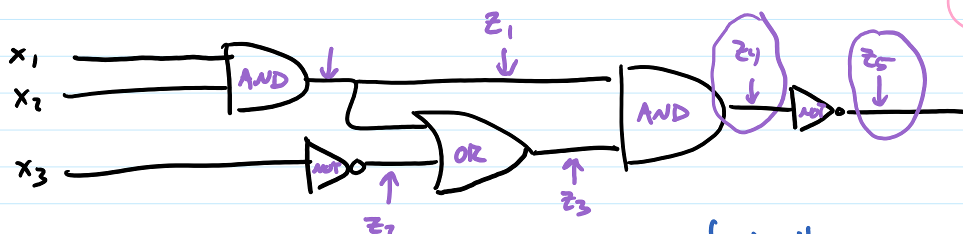


CIRCUIT-SAT(C)
create φ from C — poly
output SAT(φ)
1 call to SAT



size of label doubles with each gate
n gates → size c^n exponential :(

if $SAT \in P$ then CIRCUIT-SAT $\in P$
|||
if CIRCUIT-SAT $\notin P$ then $SAT \notin P$



polynomial in size of circuit

$$
\varphi = \begin{array}{l}
z_5 \;\wedge \\
(z_5 \longleftrightarrow \sim z_4) \wedge \\
(z_4 \longleftrightarrow z_1 \wedge z_3) \wedge \\
(z_3 \longleftrightarrow z_1 \vee z_2) \wedge \\
(z_2 \longleftrightarrow \sim x_3) \wedge \\
(z_1 \longleftrightarrow x_1 \wedge x_2)
\end{array}
$$

n gates
$n+1$ terms

each term of $O(1)$ size

SAT and 3-SAT

↪ given φ in 3CNF, determine if φ is satisfiable $\quad (x) \wedge (\ldots)$

↪ $(x \vee \sim y \vee z) \wedge (x \vee y \vee \sim z) \wedge (w \vee x \vee \sim y)$ $\quad (x \vee x \vee x) \wedge (\ldots)$

3-SAT $\in$ NP : leave to viewer

SAT $\leq_p$ 3-SAT

[goal: given φ, create 3-CNF φ' s.t. [φ is satisfiable $\Updownarrow$ φ' is satisfiable]

$\varphi' \not\approx \varphi$

$$\varphi : (x \underset{z_1}{\wedge} y) \underset{z_4}{\wedge} \underset{z_3}{\sim} (x \underset{z_2}{\vee} y)$$

$$z_4 \wedge (z_4 \leftrightarrow z_1 \wedge z_3) \wedge (z_3 \leftrightarrow \sim z_2) \wedge (z_2 \leftrightarrow x \vee y) \wedge (z_1 \leftrightarrow x \wedge y)$$

≤ 100 chars

translate each line to 3CNF using procedure below

x	y	z_1	$z_1 \leftrightarrow x \wedge y$
T	T	T	T
T	T	F	F
T	F	T	F
T	F	F	T
F	T	T	F
F	T	F	T
F	F	T	F
F	F	F	T

SAT(φ)

poly: construct φ' from φ

output 3-SAT(φ')

3-SAT $\in$ P $\rightarrow$ SAT $\in$ P

SAT $\notin$ P $\rightarrow$ 3-SAT $\notin$ P

$$z_1 \leftrightarrow x \wedge y \equiv (x \wedge y \wedge z_1) \vee (x \wedge \sim y \wedge \sim z_1) \vee (\sim x \wedge y \wedge z_1) \vee (\sim x \wedge \sim y \wedge \sim z_1)$$

wanted $(\_ \vee \_ \vee \_) \wedge (\_ \vee \_ \vee \_) \ldots$

$$\sim (z_1 \leftrightarrow x \wedge y) \equiv (x \wedge y \wedge \sim z_1) \vee (x \wedge \sim y \wedge z_1) \vee (\sim x \wedge y \wedge z_1) \vee (\sim x \wedge \sim y \wedge z_1)$$

$$z_1 \leftrightarrow x \wedge y \equiv \sim\sim (z_1 \leftrightarrow x \wedge y) \equiv \sim ((x \wedge y \wedge \sim z_1) \vee (x \wedge \sim y \wedge z_1) \vee (\sim x \wedge y \wedge z_1) \vee (\sim x \wedge \sim y \wedge z_1))$$

$$\equiv \underbrace{(\sim x \vee \sim y \vee z_1) \wedge (\sim x \vee y \vee z_1) \wedge (\ \vee\ \vee\ ) \wedge (\ \vee\ \vee\ )}_{\leq 8 \text{ clauses, each of which has 3 terms}}$$