polynomial time: $O(n^c)$

for some constant c

Circuit SAT

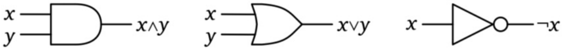
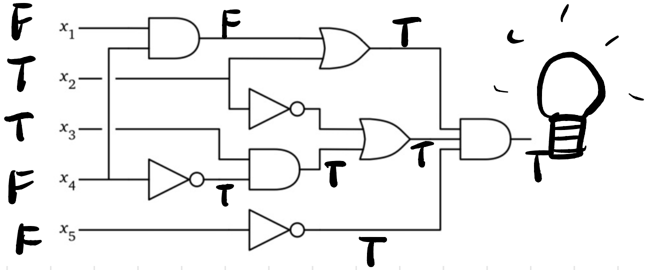


Figure 15.1. An AND gate, an OR gate, and a NOT gate.



Input: A boolean circuit with n inputs & one output wire.

Can we set the inputs so the output is True?

Easy to verify that a given set of inputs results in outputting True

O(n) time!

No fast algorithm known to check if you can turn on bulb!

Decision problem:

Output is True or False.

Three classes of decision problems:

P: Can be solved in polynomial time.

"Does the MST of G have weight $\leq k$?"

NP: <u>If the answer is True</u>, there is a proof you can verify in polynomial time.

Ex: Circuit SAT

(Cannot fool the verifier if answer is False.)

co-NP: If answer is False, can verify a proof in polynomial time.

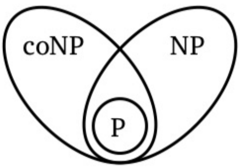
Ex: PRIME: Given an n-bit integer, is it prime?

NP: Non-deterministic polynomial

$P \subseteq NP$

Big Question: $P \overset{?}{=} NP$

One of seven
Millennium Prize
Problems

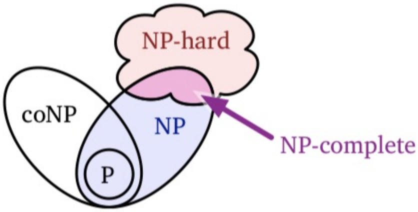


Problem B is NP-hard if we can reduce every problem A in NP to problem B in polynomial time.

$\Rightarrow$ a poly time algorithm for B implies $P = NP$

If $P \neq NP$, there is no poly time algorithm for B.

A decision problem B is NP-complete if B is NP-hard & B ∈ NP



⇒ poly time alg iff P = NP

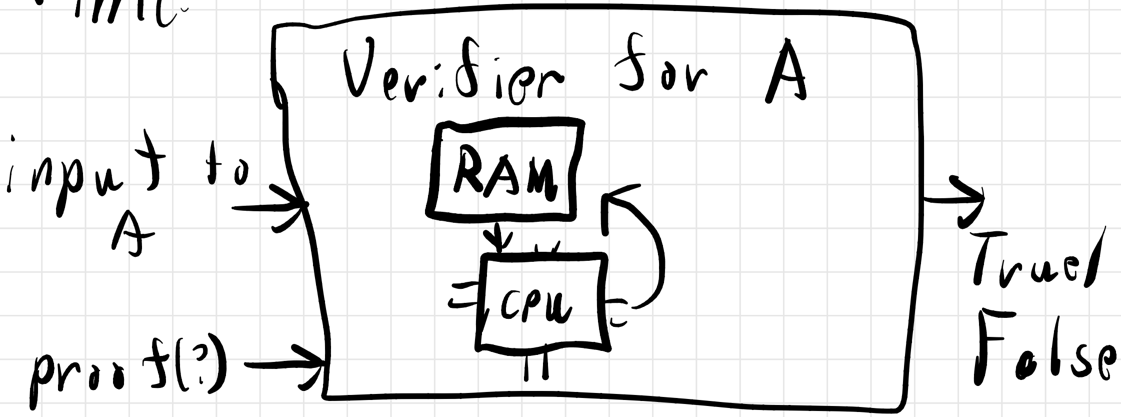
Cook ('71) & Levin ('73):

Circuit SAT is NP-complete.

"Proof":

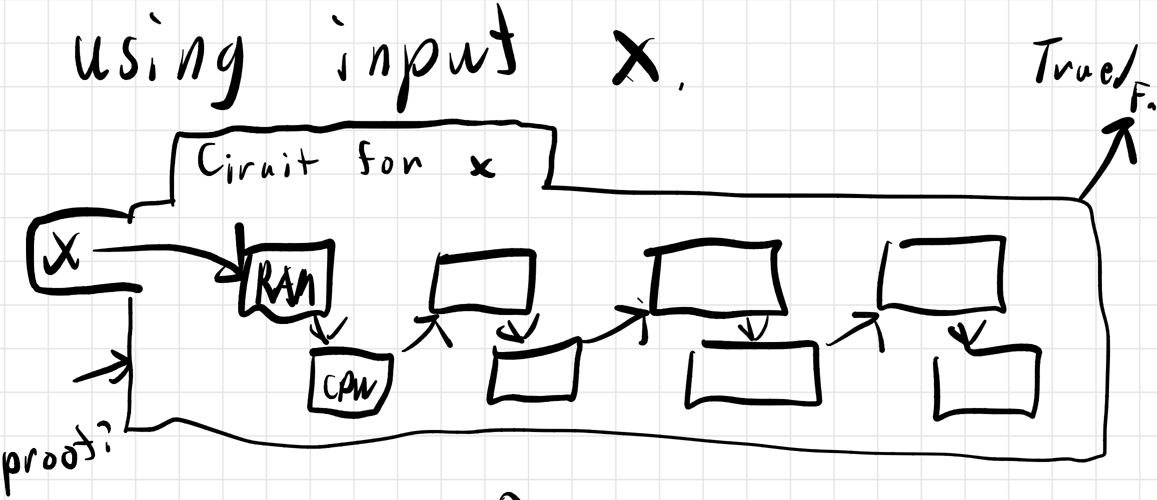
Let problem $A \in NP$.

You can verify proofs of True inputs to A in poly time



Uses polynomial amount of RAM + polynomial # clock cycles.

Say we want to solve A using input x.



copy RAM + CPU poly times

Can satisfy circuit iff there is a setting for "proof" input wires.

$\Rightarrow$ Circuit SAT is NP-hard.

We saw it $\in$ NP

$\Rightarrow$ $\in$ NP-complete.

Reduction argument

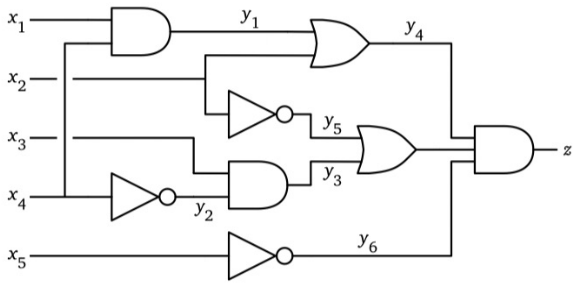
To prove B is NP-hard, reduce *a* *known* NP-hard problem A to problem B in polynomial time.

Ex: Formula satisfiability (SAT):

Given a boolean formula, can you set the variables so the formula evaluates to True?

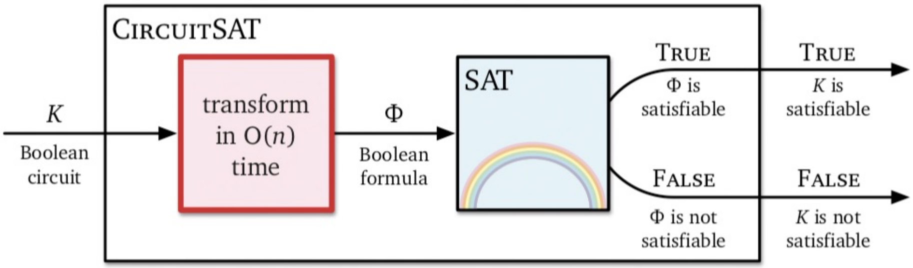
Claim: SAT is NP-complete.

Reduce <u>from</u> Circuit SAT



$$(y_1 = x_1 \wedge x_4) \wedge (y_2 = \overline{x_4}) \wedge (y_3 = x_3 \wedge y_2) \wedge (y_4 = y_1 \vee x_2) \wedge$$
$$(y_5 = \overline{x_2}) \wedge (y_6 = \overline{x_5}) \wedge (y_7 = y_3 \vee y_5) \wedge (z = y_4 \wedge y_7 \wedge y_6) \wedge z$$

- assign each wire a variable
- write a small formula describing each gate
- add the formula z just for output wire
- $\wedge$ them all together



SAT $\in$ P $\Rightarrow$ CircuitSAT $\in$ P

$\Rightarrow$ P = NP

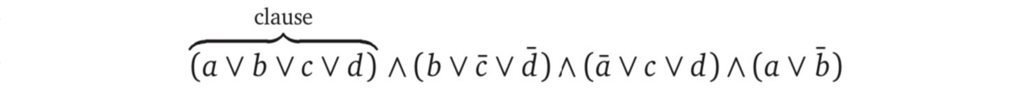
Also, SAT $\in$ NP ("proof" is variable assignments)

$\Rightarrow$ SAT is NP-complete

3SAT

- a literal is a variable or its negation (a, $\bar{a}$)
- a clause a disjunction ($\vee$) of literals
- conjunctive normal form (CNF):

  conjunction ($\wedge$ ~~AND~~) of clauses



$$\overbrace{(a \vee b \vee c \vee d)}^{\text{clause}} \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b})$$

3CNF: CNF with exactly 3 literals per clause

3SAT: SAT but input is 3CNF.

Claim: 3SAT is NP-complete.

From Circuit SAT.

1) Simplify circuit so each gate has ≤ 2 inputs.

2) Write little formulas like before.

3)

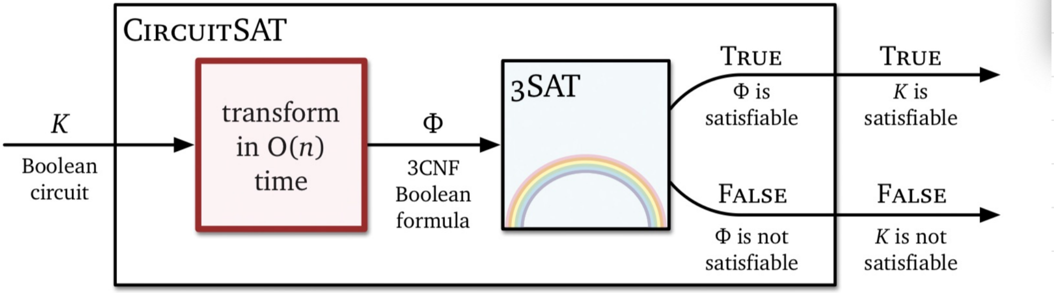
$$\begin{aligned} a = b \wedge c &\longmapsto (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee b) \wedge (\bar{a} \vee c) \\ a = b \vee c &\longmapsto (\bar{a} \vee b \vee c) \wedge (a \vee \bar{b}) \wedge (a \vee \bar{c}) \\ a = \bar{b} &\longmapsto (a \vee b) \wedge (\bar{a} \vee \bar{b}) \end{aligned}$$

4)

$$\begin{aligned} a \vee b &\longmapsto (a \vee b \vee x) \wedge (a \vee b \vee \bar{x}) \\ a &\longmapsto (a \vee x \vee y) \wedge (a \vee \bar{x} \vee y) \wedge (a \vee x \vee \bar{y}) \wedge (a \vee \bar{x} \vee \bar{y}) \end{aligned}$$

$$\begin{gathered} (y_1 \vee \overline{x_1} \vee \overline{x_4}) \wedge (\overline{y_1} \vee x_1 \vee z_1) \wedge (\overline{y_1} \vee x_1 \vee \overline{z_1}) \wedge (\overline{y_1} \vee x_4 \vee z_2) \wedge (\overline{y_1} \vee x_4 \vee \overline{z_2}) \\ \wedge (y_2 \vee x_4 \vee z_3) \wedge (y_2 \vee x_4 \vee \overline{z_3}) \wedge (\overline{y_2} \vee \overline{x_4} \vee z_4) \wedge (\overline{y_2} \vee \overline{x_4} \vee \overline{z_4}) \\ \wedge (y_3 \vee \overline{x_3} \vee \overline{y_2}) \wedge (\overline{y_3} \vee x_3 \vee z_5) \wedge (\overline{y_3} \vee x_3 \vee \overline{z_5}) \wedge (\overline{y_3} \vee y_2 \vee z_6) \wedge (\overline{y_3} \vee y_2 \vee \overline{z_6}) \\ \wedge (\overline{y_4} \vee y_1 \vee x_2) \wedge (y_4 \vee \overline{x_2} \vee z_7) \wedge (y_4 \vee \overline{x_2} \vee \overline{z_7}) \wedge (y_4 \vee \overline{y_1} \vee z_8) \wedge (y_4 \vee \overline{y_1} \vee \overline{z_8}) \\ \wedge (y_5 \vee x_2 \vee z_9) \wedge (y_5 \vee x_2 \vee \overline{z_9}) \wedge (\overline{y_5} \vee \overline{x_2} \vee z_{10}) \wedge (\overline{y_5} \vee \overline{x_2} \vee \overline{z_{10}}) \\ \wedge (y_6 \vee x_5 \vee z_{11}) \wedge (y_6 \vee x_5 \vee \overline{z_{11}}) \wedge (\overline{y_6} \vee \overline{x_5} \vee z_{12}) \wedge (\overline{y_6} \vee \overline{x_5} \vee \overline{z_{12}}) \\ \wedge (\overline{y_7} \vee y_3 \vee y_5) \wedge (y_7 \vee \overline{y_3} \vee z_{13}) \wedge (y_7 \vee \overline{y_3} \vee \overline{z_{13}}) \wedge (y_7 \vee \overline{y_5} \vee z_{14}) \wedge (y_7 \vee \overline{y_5} \vee \overline{z_{14}}) \\ \wedge (y_8 \vee \overline{y_4} \vee \overline{y_7}) \wedge (\overline{y_8} \vee y_4 \vee z_{15}) \wedge (\overline{y_8} \vee y_4 \vee \overline{z_{15}}) \wedge (\overline{y_8} \vee y_7 \vee z_{16}) \wedge (\overline{y_8} \vee y_7 \vee \overline{z_{16}}) \\ \wedge (y_9 \vee \overline{y_8} \vee \overline{y_6}) \wedge (\overline{y_9} \vee y_8 \vee z_{17}) \wedge (\overline{y_9} \vee y_6 \vee z_{18}) \wedge (\overline{y_9} \vee y_6 \vee \overline{z_{18}}) \wedge (\overline{y_9} \vee y_8 \vee \overline{z_{17}}) \\ \wedge (y_9 \vee z_{19} \vee z_{20}) \wedge (y_9 \vee \overline{z_{19}} \vee z_{20}) \wedge (y_9 \vee z_{19} \vee \overline{z_{20}}) \wedge (y_9 \vee \overline{z_{19}} \vee \overline{z_{20}}) \end{gathered}$$

(our favorite circuit)


CIRCUITSAT
K
Boolean circuit
transform in O(n) time
Φ
3CNF Boolean formula
3SAT
TRUE
Φ is satisfiable
FALSE
Φ is not satisfiable
TRUE
K is satisfiable
FALSE
K is not satisfiable