"Efficient" ≡ polynomial time
($O(n^c)$ for some constant c)

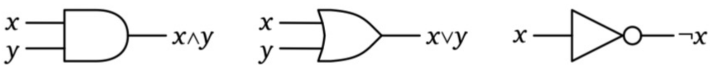
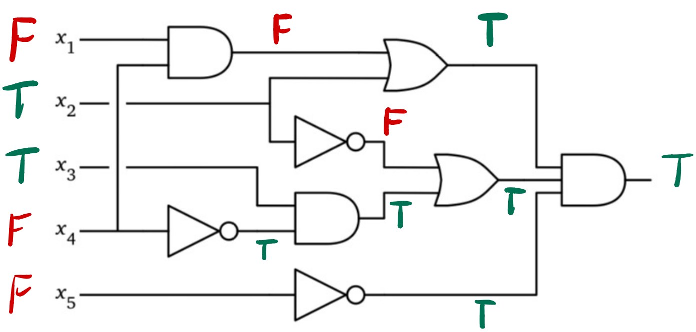


Figure 15.1. An AND gate, an OR gate, and a NOT gate.



Can you set $x_1, \ldots, x_5$ to True or False so circuit outputs True?
(yes, for this example)

<u>circuit satisfiability</u> (Circuit SAT)

Easy verify a "Yes" answer
given the correct assignments.
Hard to solve from scratch:
try all $O(2^n)$ inputs?

Decision Problems: Any kind of (finite) input, output True or False (Yes or No) (1 or 0)

Three main classes (for 6363):

Polynomial

P: Decision problems with poly time algorithms.
(Given G & k, does G have a spanning tree of weight $\leq k$?)

Non-deterministic

NP: Decision problems where True instances have "proofs" that can be verified in poly time.
(Circuit SAT)
$NP \supseteq P$

co-NP: Decision problems where <u>False</u> instances have a "disproof" that can be verified in poly time.

(Given n-bit number X, is X prime?)

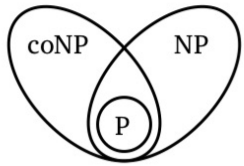
actually in P!

$P \subseteq NP$ (if $A \in P$, always use "empty proof")

But, does $P = NP$?

I think $P \neq NP$ ($P \subsetneq NP$)

Another problem: does NP = co-NP?



"The world?"
(no proof)

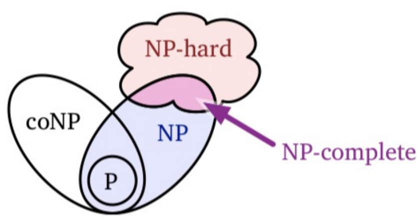
Probem B (decision or not) is **NP-hard** if we can reduce every problem $A \in NP$ to an algorithm for B with poly time overhead.

$\Rightarrow$ Any poly time algorithm for B yields poly time algs for all $A \in NP$.

$\Rightarrow$ (Poly time alg for B
$\Rightarrow$ P = NP)

$\Rightarrow$ no poly time alg for B, probably.

NP-complete: in NP and NP-hard



(HALT is NP-hard but not in NP)

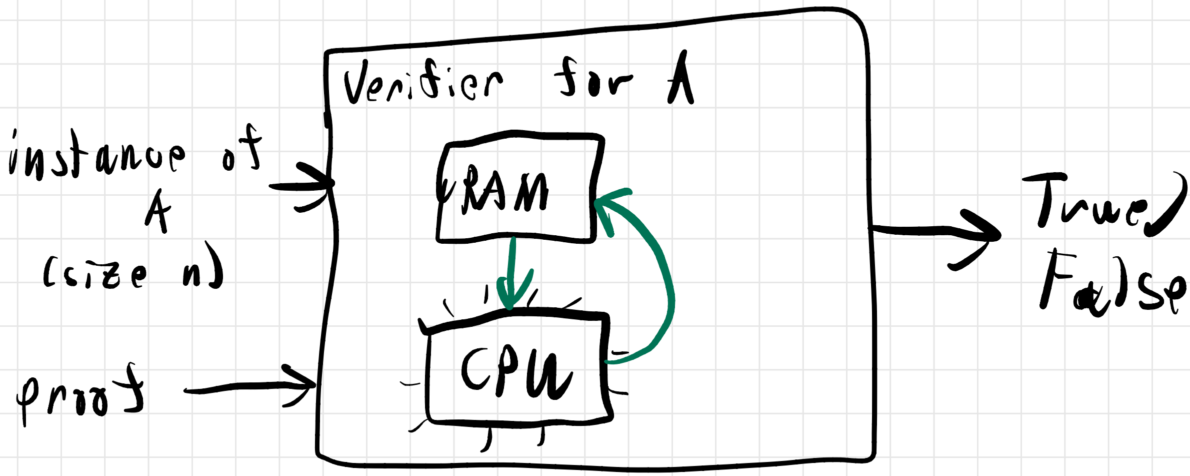
Each NP-complete problem is in P iff P = NP.

Cook ['71], Levin ['73]:

Circuit SAT is NP-complete.

$A \in NP$.

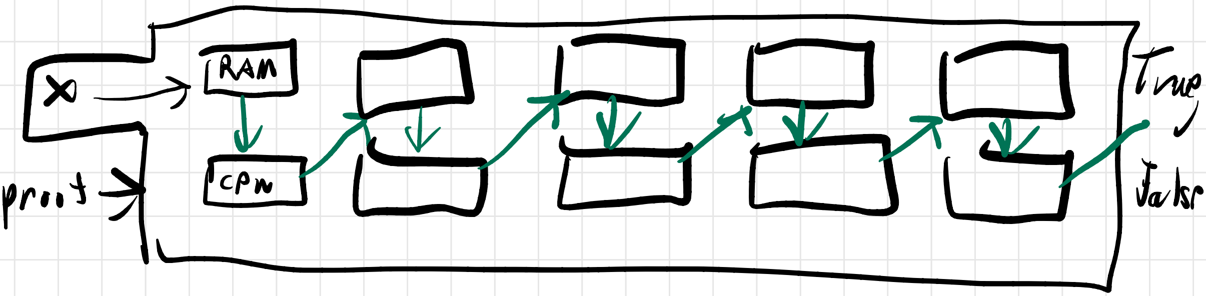
So we can verify True proofs in poly time.



uses poly(n) clock cycles

uses poly(n) bits of RAM

Suppose we're given instance x of A. $n := |x|$.



a giant boolean circuit of size poly(n)

(an instance of CircuitSAT!)

answer for x is True iff $\exists$ a proof to be verified iff answer for CircuitSAT instance is True so Circuit SAT is NP-hard.

Reduction Arguments:

To prove problem B is NP-hard, take a <u>known</u> NP-hard problem A & reduce A to B with poly time overhead.

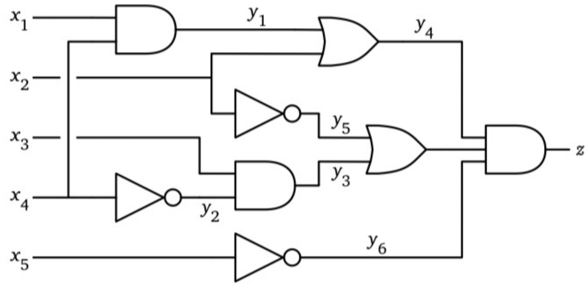
formula satisfiability (SAT)

Given boolean formula of any form. Can you set the variables so whole formula is true?

Reduce <u>from</u> CircuitSAT

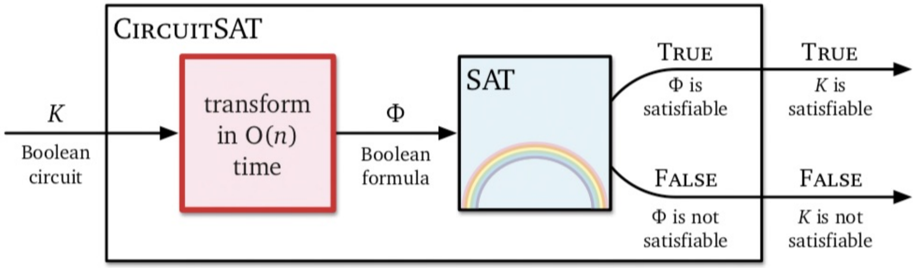
Given a circuit...

- add a new variable for the output of each gate
- write a big conjunction of equations describing each gate
- add output of final gate to end of conjunction



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

formula is sat. iff circuit is



Time for CircuitSAT is $O(n)$ + time for SAT

$\Rightarrow$ SAT is NP-hard

SAT $\in$ NP (show me how to set the variables)

so SAT is NP-complete!