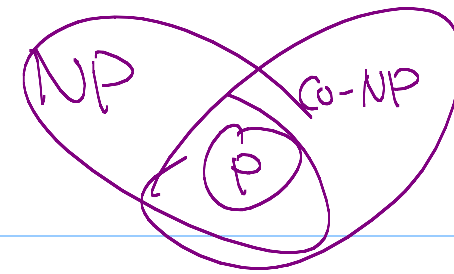
CS314 - Reductions

Note Title 3/24/2010

Announcements

- Next HW up today / tomorrow
  written, due ~~next Friday~~ at beginning of class
  Sometime

  (probably Easter Wed.)



P, NP, & co-NP

Consider decision problems – output is a single boolean (yes or no).

Define:

- P : the set of problems that can be solved in polynomial time

nondeterministic polynomial

- NP: the set of decision problems where if the answer is Yes, there is a proof of this that can be checked in polynomial time

- co-NP: If answer is No, that can be checked in polynomial time.

NP-Hard

Dfn: A problem Π is NP-Hard
$\Longleftrightarrow$ (if & only if)
if Π can be solved in polynomial time, then $P = NP$.

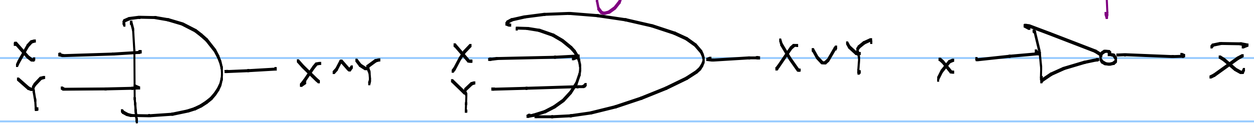
Dfn: A problem is NP-Complete if it is both NP-Hard and in NP.

These are the "hardest" problems in NP.

Circuit-SAT

Input: boolean circuit, with T/F inputs
Output: T or F

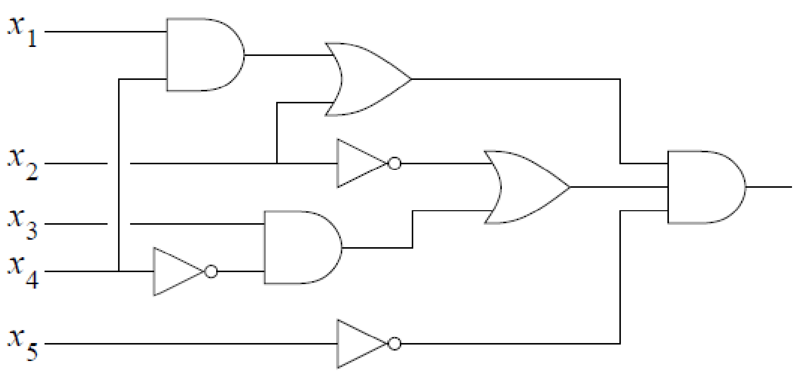
Q: Is there a set of assignments to the inputs so that output = T?



AND gate   OR gate   NOT gate

Ex:

5 inputs



1 output

A boolean circuit. inputs enter from the left, and the output leaves to the right.

Cook-Levin Theorem:

Circuit-satisfiability is NP-Complete.

Comments:

The proof is amazing - takes <u>any</u> NP problem & changes it into a circuit with polynomial size, so that:

true answer for NP problem exists

$\Updownarrow$

the resulting circuit is satisfiable

However:

This is pretty much the only direct proof to ever show a problem is NP-Complete!

So how do we say other problems are just as hard?

Reductions

Dfn: $Y \leq_p X$ (read Y is polynomial time reducable to X)

if Y can be solved using a polynomial number of steps plus a polynomial number of calls to an algorithm (or "black box") that solves X.

Ex: test question about near trees!

Found cycles

Ex: sorting

Thm: Spps $Y \le_p X$. If X can be solved in polynomial time, then so can Y.

Pf: Replace "black box" to solve X with code that runs in poly. time.

Solving Y now takes
poly time + (Poly # of calls to X) * (running time for X)

= poly. time

$\blacksquare$

This is useful for algorithms!

But what if we don't know of a polynomial time algorithm?

NP-Complete?

$p \to q$
$\Downarrow$
$\neg q \to \neg p$ (contrapositive)

Spps $Y \leq_p X$. If X can be solved in polynomial time, then so can Y.

Take the contrapositive!

Spps $Y \leq_p X$. If Y cannot be solved in polynomial time, then X can't be solved in polynomial time.

So if we take a "hard" problem &
reduce it to another problem X,
then X must be at least as hard.

Useful!

$\boxed{\text{If we want to show a problem is NP-Hard, reduce a known NP-Hard problem to it!}}$

Important!!

Ex: SAT

no $\forall, \exists$

→ Input: boolean formula

Q: Can we assign boolean values to the variables so that the formula evaluates to true?

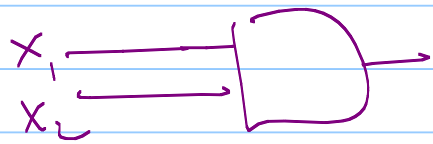
Ex:

→ $(a \vee b \vee c \vee \bar{d}) \Leftrightarrow ((b \wedge \bar{c}) \vee \overline{(\bar{a} \Rightarrow d)} \vee (c \neq a \wedge b)),$

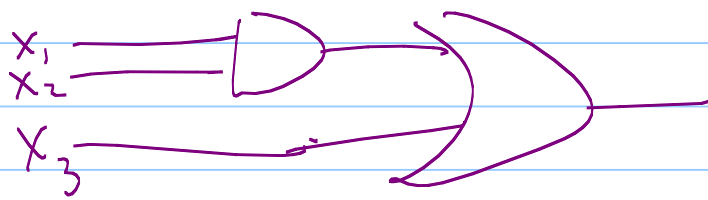
Show SAT is NP-Hard.

How to show NP-Hard?

Reduce circuit SAT to SAT

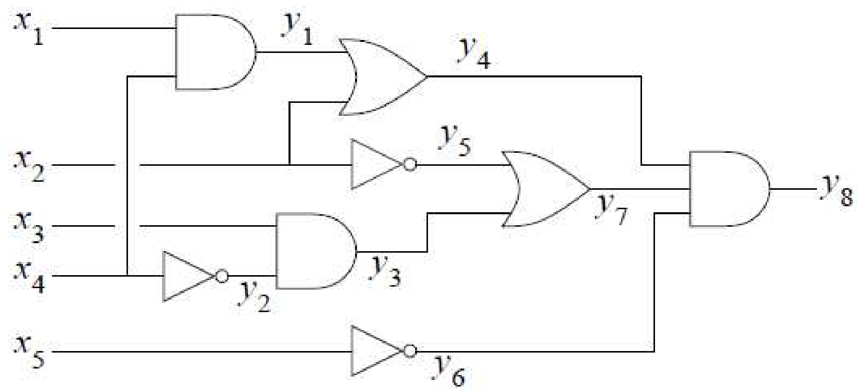


$\Longleftrightarrow \quad x_1 \wedge x_2$



$\Longleftrightarrow \quad (x_1 \wedge x_2) \vee x_3$

Ex:



$$(y_1 = x_1 \wedge x_4) \wedge (y_2 = \overline{x_4}) \wedge (y_3 = x_3 \wedge y_2) \wedge (y_4 = y_1 \vee x_2) \wedge$$
$$(y_5 = \overline{x_2}) \wedge (y_6 = \overline{x_5}) \wedge (y_7 = y_3 \vee y_5) \wedge (y_8 = y_4 \wedge y_7 \wedge y_6) \wedge y_8$$

Alg: BFS through circuit

So in $O(n)$, I can transform circuit into a boolean formula.

But careful! Need conversion to be polynomial time, + size to stay polynomial.

Well, said takes O(n) to transform.

Every gate gives us 1 clause in the output formula.

Reduction Picture

CircuitSAT $\le_P$ SAT

$$\begin{tikzcd} \boxed{\text{boolean circuit}} \arrow[r, "O(n)"] & \boxed{\text{boolean formula}} \arrow[d, Rightarrow, "\text{SAT}"] \\ \boxed{\textsc{True or False}} & \boxed{\textsc{True or False}} \arrow[l, "\text{trivial}"'] \end{tikzcd}$$

$$T_{\text{CSAT}}(n) \le O(n) + T_{\text{SAT}}(O(n)) \implies T_{\text{SAT}}(n) \ge T_{\text{CSAT}}(\Omega(n)) - O(n)$$

circuit is satisfiable $\iff$ boolean formula is satisfiable

Another example: 3SAT

Dfn: A boolean formula is in conjunctive normal form (CNF) if it is a conjunction (or AND) of clauses, each of which is a disjunction (or OR) of variables or negations.

Ex:

$$\underbrace{(a \vee b \vee c \vee d)}_{\text{clause of variables or-ed together}} \wedge \underbrace{(b \vee \bar{c} \vee \bar{d})} \wedge \underbrace{(\bar{a} \vee b)}$$

"and" the clauses together

3SAT (cont)

$$\underbrace{(a \vee b \vee c)}_{\text{exactly 3 variables in each clause}} \wedge (\bar{d} \vee \bar{a} \vee b) \wedge \ldots$$

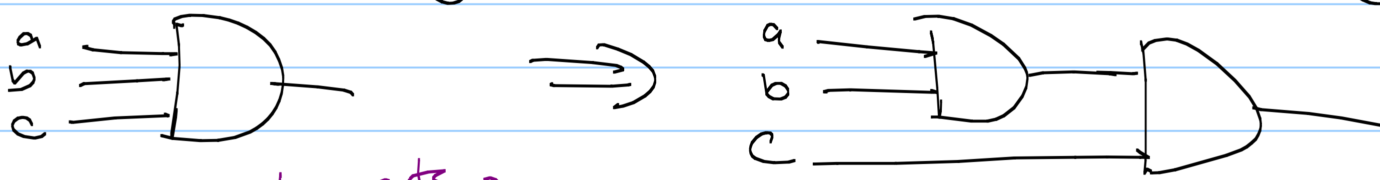
Dfn: A 3CNF formula has exactly 3 variables per clause.

3SAT: Given a 3CNF formula, is the an assignmenment of variables which makes the formula evaluate to true?

How do we show NP-Complete?
Reduce Circuit SAT or SAT to 3SAT.

Reduction <u>from</u> CircuitSAT:

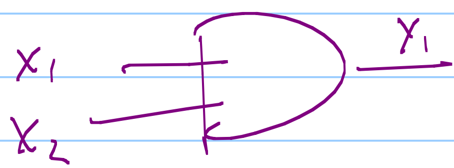
(1) Make every and/or gate have only 2 inputs.



k inputs →

(replace with binary tree of $k-1$ 2input gates)

(2) Write down clause for each gate
(same as before)



$(y_1 = x_1 \wedge x_2)$

not CNF

(3) Change each clause to CNF formula:

$$a = b \wedge c \longmapsto (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee b) \wedge (\bar{a} \vee c)$$

$$a = b \vee c \longmapsto (\bar{a} \vee b \vee c) \wedge (a \vee \bar{b}) \wedge (a \vee \bar{c})$$

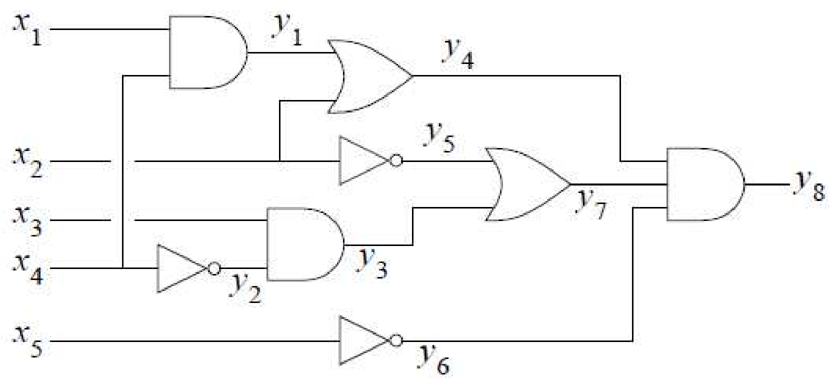
$$a = \bar{b} \longmapsto (a \vee b) \wedge (\bar{a} \vee \bar{b})$$

(4) Make sure every clause has 3 literals:

$$a \longmapsto (a \vee x \vee y) \wedge (a \vee \bar{x} \vee y) \wedge (a \vee x \vee \bar{y}) \wedge (a \vee \bar{x} \vee \bar{y})$$

$$a \vee b \longmapsto (a \vee b \vee x) \wedge (a \vee b \vee \bar{x})$$

Ex:



⇒

$$
\begin{aligned}
&(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_8 \vee \overline{z_{17}}) \wedge (\overline{y_9} \vee y_6 \vee z_{18}) \wedge (\overline{y_9} \vee y_6 \vee \overline{z_{18}}) \\
\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{aligned}
$$

Looks huge!

But: • every gate became at most (k-1) gates, where k = # of inputs

• Every gate then formed at most 5 clauses

So transformation & size are both polynomial.

O(n) algorithm &
O(n) size boolean in 3CNF form.

Recap:

In 3CNF form!

$$
\begin{tikzcd}
\boxed{\text{boolean circuit}} \arrow[r, "O(n)"] & \boxed{\text{boolean formula}} \arrow[d, Rightarrow, "\text{SAT}"] \\
\boxed{\text{True or False}} & \boxed{\text{True or False}} \arrow[l, "\text{trivial}"']
\end{tikzcd}
$$

$$T_{\mathrm{CSAT}}(n) \leq O(n) + T_{\mathrm{SAT}}(O(n)) \quad \Longrightarrow \quad T_{\mathrm{SAT}}(n) \geq T_{\mathrm{CSAT}}(\Omega(n)) - O(n)$$

Circuit is satisfiable $\Longleftrightarrow$ 3CNF formula is satisfiable

So 3SAT is NP-Complete.

<u>Next time</u>: non-logic reductions

(I promise!)