CPSC 420 Lecture 18: Today's announcements:

- HW3 is on Gradescope, due Mar 9, 23:59
- Examlet 3 on Mar 17 in class. Closed book & no notes
- Reading:
  NP-hardness [by Erickson]

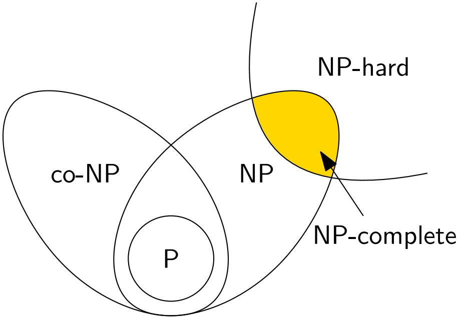
Today's Plan

- NP-hardness

NP-hard, NP-complete

NP-hard: Problems A where if A can be solved in polynomial time then $P = NP$.

NP-complete: Decision problems A where $A \in$ NP-hard and $A \in$ NP.



Our current guess of P, NP, co-NP, NP-hard, NP-complete

Cook-Levin Theorem

Theorem

Circuit satisfiability is NP-hard

Proof.

You don't need to know the proof but the idea is: Show how to encode the execution of any polynomial-time, non-deterministic Turing machine M on an input x as some boolean circuit that is satisfiable if and only if M outputs "Yes" on input x. □

How do we show other problems are NP-hard?

Cook-Levin Theorem

Theorem

Circuit satisfiability is NP-hard

Proof.

You don't need to know the proof but the idea is: Show how to encode the execution of any polynomial-time, non-deterministic Turing machine M on an input x as some boolean circuit that is satisfiable if and only if M outputs "Yes" on input x. □

How do we show other problems are NP-hard?

> To prove problem A is NP-hard, show how to **reduce** (in polynomial time) an NP-hard problem to A.

Reduce Circuit Satisfiability to A means "Solve Circuit Satisfiability using an algorithm for A."

3SAT is NP-hard

3 literals per clause

literals

variables a, b, c, d

3SAT: Is a given boolean formula, which is the AND of several three-literal OR-clauses, satisfiable?

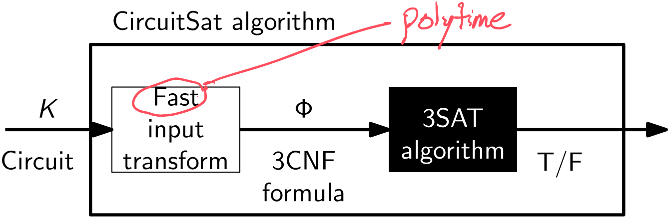
3CNF = 3 Conjunctive Normal Form

Example: Is $\underbrace{(a \vee \overline{b} \vee c)}_{\text{clause}} \wedge (\overline{a} \vee \overline{b} \vee \overline{c}) \wedge (a \vee \overline{c} \vee \overline{d})$ satisfiable?

a = T

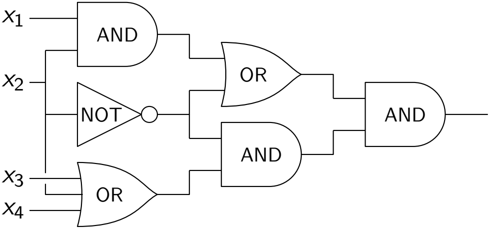
b = F

Reduction: From CircuitSat to 3SAT



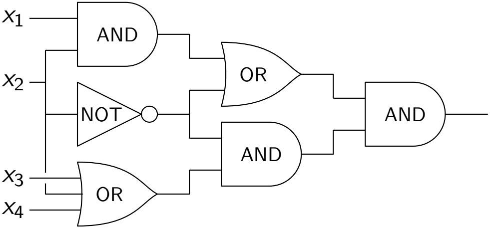
Transform Circuit into 3CNF formula

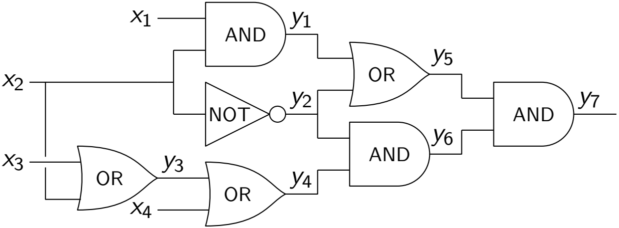
Input to CircuitSAT = Boolean Circuit



Transform Circuit into 3CNF formula

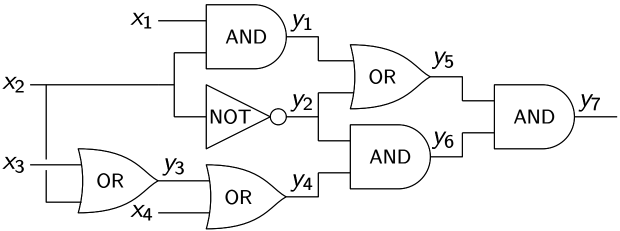
Input to CircuitSAT = Boolean Circuit

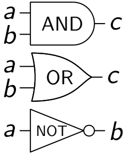




Equivalent circuit where AND and OR gates have two inputs.

Transform Circuit into 3CNF formula





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

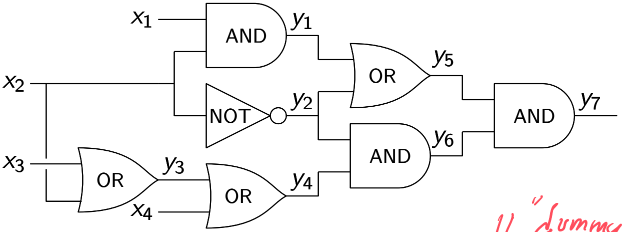
$\longmapsto (a \vee b \vee \overline{c}) \wedge (\overline{a} \vee c) \wedge (\overline{b} \vee c)$

$\longmapsto (a \vee b) \wedge (\overline{a} \vee \overline{b})$

$(\overline{x_1} \vee \overline{x_2} \vee y_1) \wedge (x_1 \vee \overline{y_1}) \wedge (x_2 \vee \overline{y_1}) \wedge (x_2 \vee y_2) \wedge (\overline{x_2} \vee \overline{y_2}) \wedge (x_3 \vee x_2 \vee \overline{y_3}) \wedge (\overline{x_3} \vee y_3) \wedge (\overline{x_2} \vee y_3) \wedge (y_3 \vee x_4 \vee \overline{y_4}) \wedge (\overline{y_3} \vee y_4) \wedge (\overline{x_4} \vee y_4) \wedge (y_1 \vee y_2 \vee \overline{y_5}) \wedge (\overline{y_1} \vee y_5) \wedge (\overline{y_2} \vee y_5) \wedge (\overline{y_2} \vee \overline{y_4} \vee y_6) \wedge (y_2 \vee \overline{y_6}) \wedge (y_4 \vee \overline{y_6}) \wedge (\overline{y_5} \vee \overline{y_6} \vee y_7) \wedge (y_5 \vee \overline{y_7}) \wedge (y_6 \vee \overline{y_7}) \wedge (y_7)$? the output must be True

CNF formula satisfied iff circuit is satisfied.

Transform Circuit into 3CNF formula



$$(a \vee b) \longmapsto (a \vee b \vee z) \wedge (a \vee b \vee \overline{z})$$ add "dummy" variable z

$$(a) \longmapsto (a \vee w \vee z) \wedge (a \vee \overline{w} \vee z) \wedge (a \vee w \vee \overline{z}) \wedge (a \vee \overline{w} \vee \overline{z}))$$

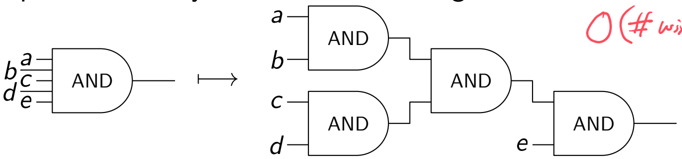
$$(\overline{x_1} \vee \overline{x_2} \vee y_1) \wedge (x_1 \vee \overline{y_1} \vee z_1) \wedge (x_1 \vee \overline{y_1} \vee \overline{z_1}) \wedge (x_2 \vee \overline{y_1} \vee z_2) \wedge (x_2 \vee \overline{y_1} \vee$$
$$\overline{z_2}) \wedge (x_2 \vee y_2 \vee z_3) \wedge (x_2 \vee y_2 \vee \overline{z_3})(\overline{x_2} \vee \overline{y_2} \vee z_4) \wedge (\overline{x_2} \vee \overline{y_2} \vee \overline{z_4}) \wedge (x_3 \vee x_2 \vee$$
$$\overline{y_3}) \wedge (\overline{x_3} \vee y_3 \vee z_5) \wedge (\overline{x_3} \vee y_3 \vee \overline{z_5}) \wedge (\overline{x_2} \vee y_3 \vee z_6) \wedge (\overline{x_2} \vee y_3 \vee \overline{z_6}) \wedge (y_3 \vee x_4 \vee$$
$$\overline{y_4}) \wedge (\overline{y_3} \vee y_4 \vee z_7) \wedge (\overline{y_3} \vee y_4 \vee \overline{z_7}) \wedge (\overline{x_4} \vee y_4 \vee z_8) \wedge (\overline{x_4} \vee y_4 \vee \overline{z_8}) \wedge (y_1 \vee y_2 \vee$$
$$\overline{y_5}) \wedge (\overline{y_1} \vee y_5 \vee z_9) \wedge (\overline{y_1} \vee y_5 \vee \overline{z_9}) \wedge (\overline{y_2} \vee y_5 \vee z_{10}) \wedge (\overline{y_2} \vee y_5 \vee \overline{z_{10}}) \wedge (\overline{y_2} \vee$$
$$\overline{y_4} \vee y_6) \wedge (y_2 \vee \overline{y_6} \vee z_{11}) \wedge (y_2 \vee \overline{y_6} \vee \overline{z_{11}}) \wedge (y_4 \vee \overline{y_6} \vee z_{12}) \wedge (y_4 \vee \overline{y_6} \vee \overline{z_{12}}) \wedge$$
$$(\overline{y_5} \vee \overline{y_6} \vee y_7) \wedge (y_5 \vee \overline{y_7} \vee z_{13}) \wedge (y_5 \vee \overline{y_7} \vee \overline{z_{13}}) \wedge (y_6 \vee \overline{y_7} \vee z_{14}) \wedge (y_6 \vee \overline{y_7} \vee$$
$$\overline{z_{14}}) \wedge (y_7 \vee z_{15} \vee z_{16}) \wedge (y_7 \vee \overline{z_{15}} \vee z_{16}) \wedge (y_7 \vee z_{15} \vee \overline{z_{16}}) \wedge (y_7 \vee \overline{z_{15}} \vee \overline{z_{16}})$$

3CNF formula satisfied iff circuit is satisfied.

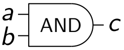
Proof: Φ is satisfiable iff K is satisfiable

1. Replace AND gates that have $k > 2$ inputs with a logically equivalent binary tree of $k-1$ AND gates. Same for OR.

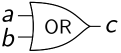
   $O(\#\text{ wires in } K)$



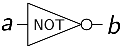
2. Add new variables y_i to each gate output. $O(\#\text{wires in } K)$
3. Use



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



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



$\longmapsto (a \vee b) \wedge (\overline{a} \vee \overline{b})$

$O(\#\text{ gates after step 1})$
$= O(\#\text{ wires in } K)$

to ensure y_i = gate output for each AND, OR, NOT gate.

Proof: Φ is satisfiable iff K is satisfiable

4. Use $O(\#\text{clauses}) = O(\#\text{wires})$

$$(a \vee b) \longmapsto (a \vee b \vee z) \wedge (a \vee b \vee \overline{z})$$

$$(a) \longmapsto (a \vee w \vee z) \wedge (a \vee \overline{w} \vee z) \wedge (a \vee w \vee \overline{z}) \wedge (a \vee \overline{w} \vee \overline{z}))$$

to convert two- or one-literal clauses into three-literal clauses.

Claim: Circuit K is satisfiable if and only if the resulting formula Φ is satisfiable.

Proof: K' with 2-input gates is logically equivalent to K
$\Rightarrow$ If an input $x_1, \ldots, x_n$ satisfies circuit K', let y_j be the output of jth gate in K' on this input and let z_k be 0 or 1 for all k (it doesn't matter). This assignment satisfies Φ.

$\Leftarrow$ If Φ has a satisfying assignment to its variables (x_is, y_js, and z_ks), the assignment to $x_1, \ldots, x_n$ satisfies circuit K.

3SAT is NP-complete

The previous reduction (from CircuitSat) proves that 3SAT is NP-hard.

To show 3SAT is NP-complete we need to show 3SAT is in NP.

3SAT is NP-complete

The previous reduction (from CircuitSat) proves that 3SAT is NP-hard.

To show 3SAT is NP-complete we need to show 3SAT is in NP.

A truth-assignment to the variables of a 3SAT formula that satisfies the formula (a proof of a "Yes" instance) can be checked in linear time by checking that each clause evaluates to True.

3SAT is NP-complete

The previous reduction (from CircuitSat) proves that 3SAT is NP-hard.

To show 3SAT is NP-complete we need to show 3SAT is in NP.

A truth-assignment to the variables of a 3SAT formula that satisfies the formula (a proof of a “Yes” instance) can be checked in linear time by checking that each clause evaluates to True.

Is SAT (clauses can have any number of literals) NP-complete?

CNF

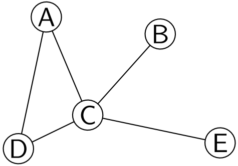
yes

SAT $\in$ NP

$\Rightarrow$ SAT is NP-complete

Independent Set

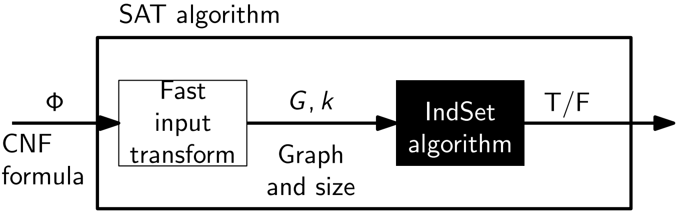
An **independent set** is a set of vertices in a graph G that share no common edge.



IndependentSet takes graph G and integer k and outputs "Yes" if G has an independent set of size k and "No" otherwise.

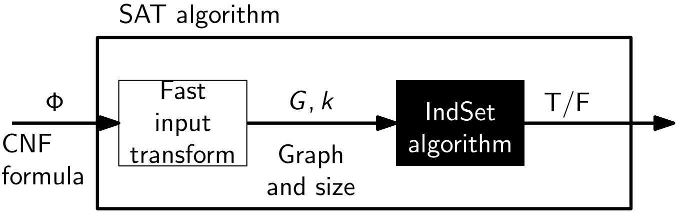
Claim: IndependentSet is NP-hard.

Reduce from SAT to IndependentSet



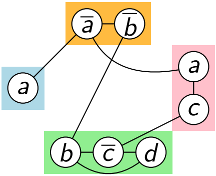
Transform a 3CNF formula Φ into a graph G and integer k so that Φ is satisfied if and only if G has an independent set of size k.

Reduce from SAT to IndependentSet



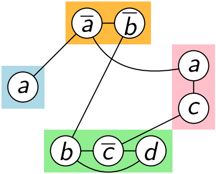
Transform a 3CNF formula Φ into a graph G and integer k so that Φ is satisfied if and only if G has an independent set of size k.

$$\Phi = (a) \wedge (\overline{a} \vee \overline{b}) \wedge (a \vee c) \wedge (b \vee \overline{c} \vee d)$$



Reduce from SAT to IndependentSet

$$\Phi = (a) \wedge (\overline{a} \vee \overline{b}) \wedge (a \vee c) \wedge (b \vee \overline{c} \vee d)$$

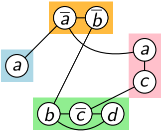
$G =$ 

$k = 4$

1. Create a vertex for every occurrence of a literal in a clause.
2. Create edges between every literal occurrence and its negation.
3. For each clause, create edges between all literals in the clause.
4. Let the size of the desired independent set $k = \#$ clauses

Reduce from SAT to IndependentSet

$$\Phi = (a) \wedge (\overline{a} \vee \overline{b}) \wedge (a \vee c) \wedge (b \vee \overline{c} \vee d)$$

$G =$  $k = 4$

Claim: G contains an independent set of size k if and only if Φ is satisfiable.

$\Rightarrow$ Let S be an independent set of size k in G. S cannot contain two literal nodes from the same clause, so every one of the k clauses contains one literal in S. S cannot contain a literal node and its negation. Set all literals in S to true. This satisfies Φ.

$\Leftarrow$ Let A be a truth assignment satisfying Φ. Every clause contains at least one True literal. Pick one for each of the k clauses and let S be the set of corresponding vertices. Since A doesn't assign True to a literal and its negation, S is an independent set of size k.