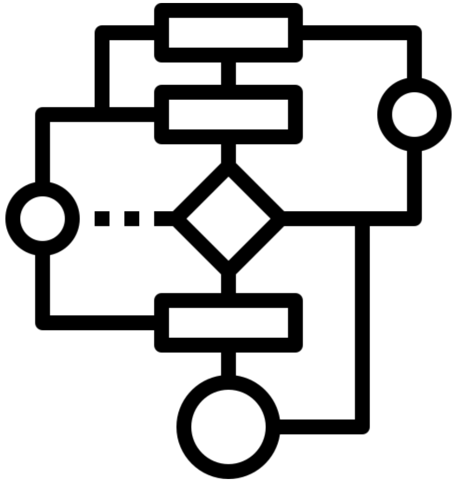
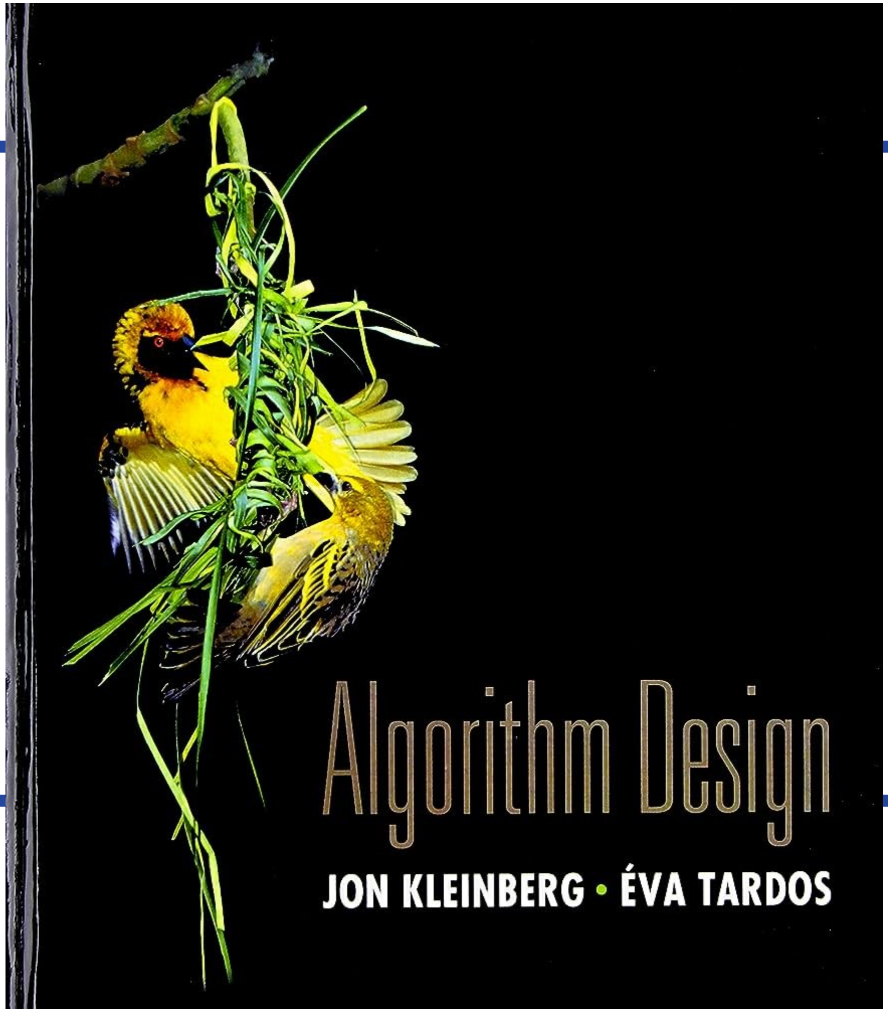


CS 310: Algorithms

Lecture 25

Instructor: Naveed Anwar Bhatti

Few Slides taken from Dr. Imdad's CS 510 course



Chapter 8:
NP and Computational Intractability

Section 8.3 :
NP-hard and NP-Complete

NP-Complete vs NP-Hard

A problem X is $\mathrm{NP\text{-}H\small{ARD}}$, if every problem in NP is polynomial time reducible to X

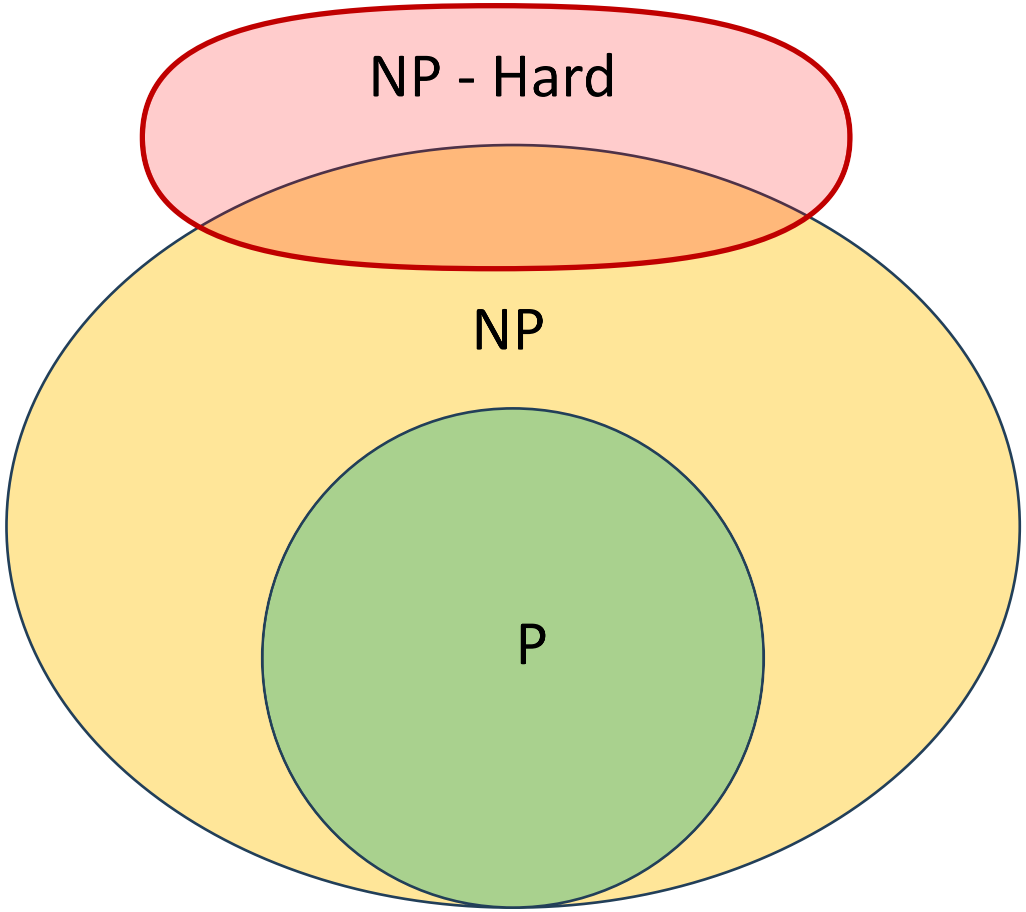
$$\forall\, Y \in \mathrm{NP}, \quad Y \ \leq_p \ X$$

A problem $X \in \mathrm{NP}$ is $\mathrm{NP\text{-}C\small{OMPLETE}}$, if every problem in NP is polynomial time reducible to X

$$X \in \mathrm{NP} \quad \text{AND} \quad \forall\, Y \in \mathrm{NP}, \quad Y \ \leq_p \ X$$

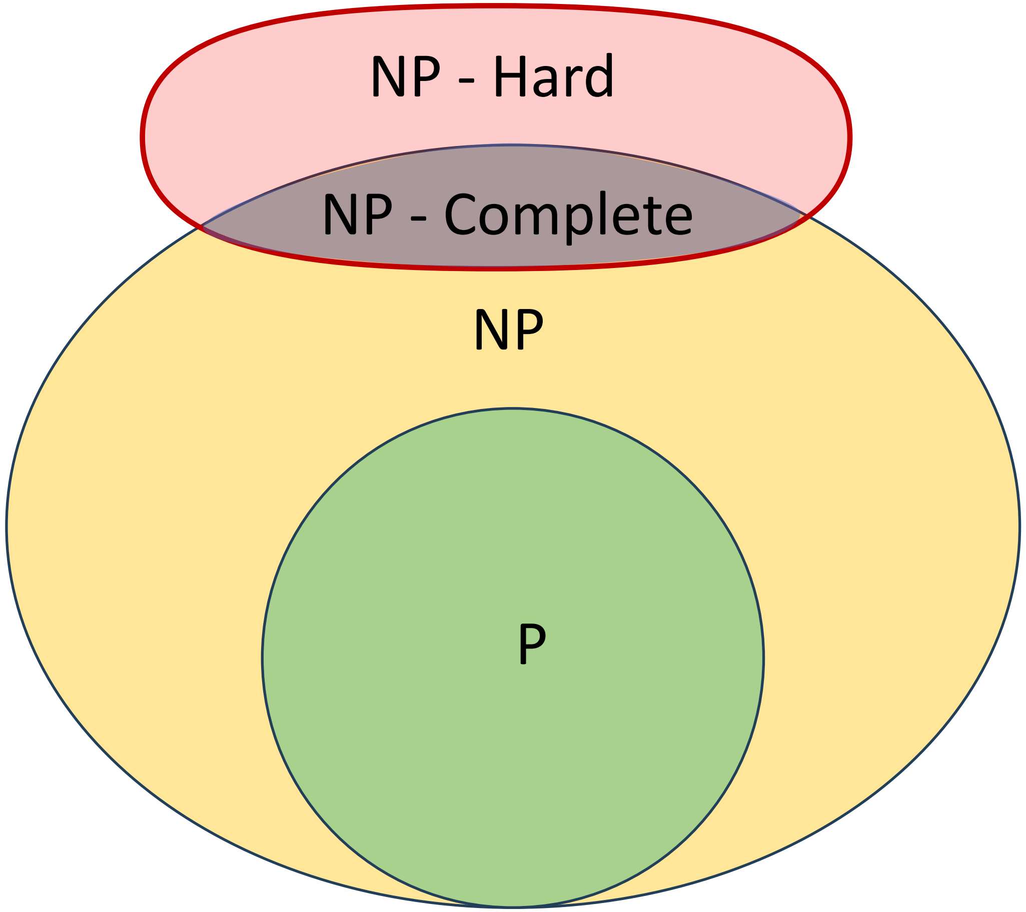
NP-Complete vs NP-Hard

A problem X is NP-HARD, if every problem in NP is polynomial time reducible to X



NP-Complete vs NP-Hard

A problem $X \in \mathrm{NP}$ is NP-COMPLETE, if every problem in NP is polynomial time reducible to X

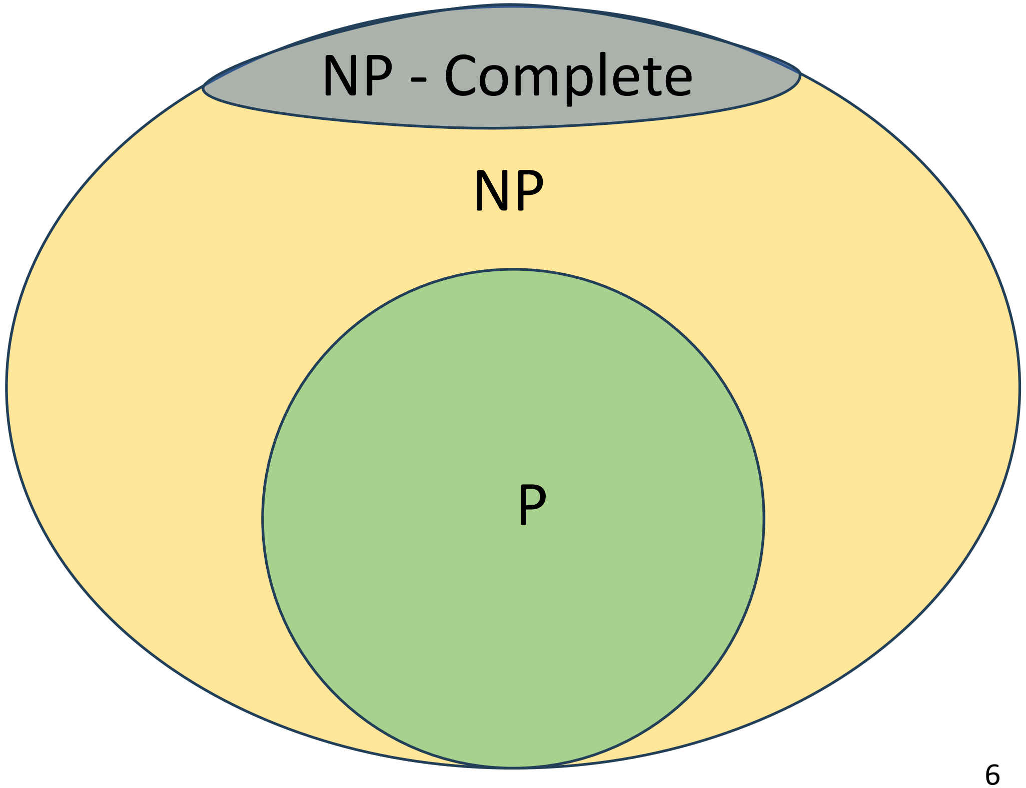


NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\ Y \leq_p X$

$\mathrm{P} \subseteq \mathrm{NP}$

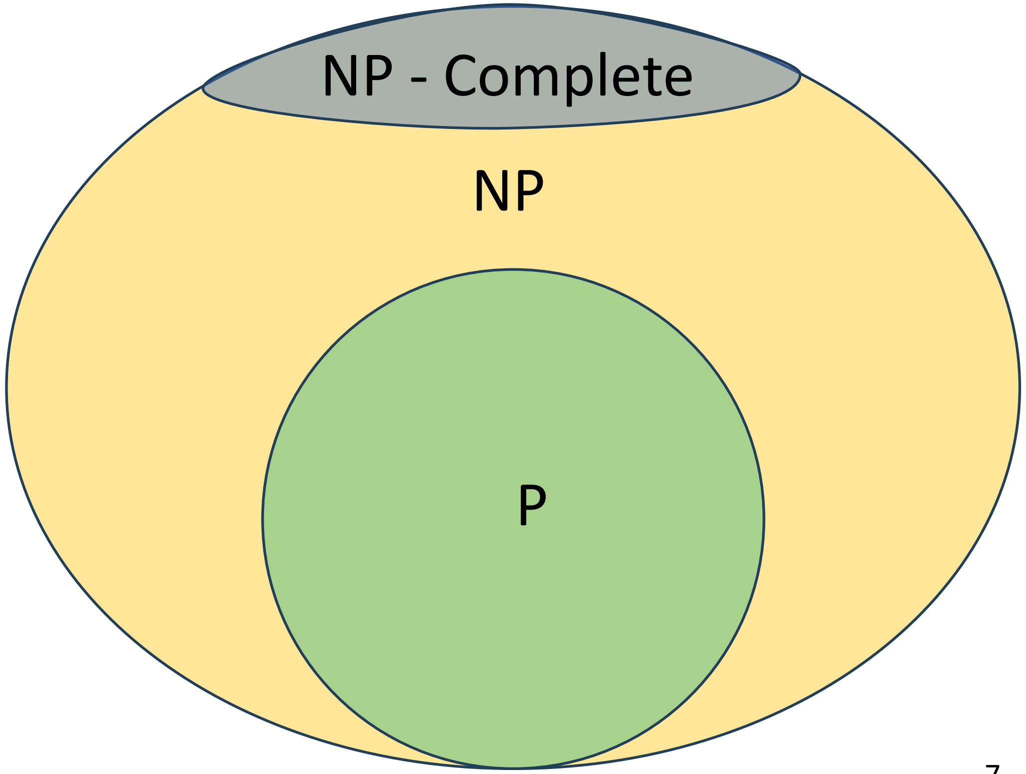


NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\;\; Y \leq_p X$

$\mathrm{P} \subseteq \mathrm{NP}$ $\quad\quad$ $\mathrm{NPC} \subseteq \mathrm{NP}$



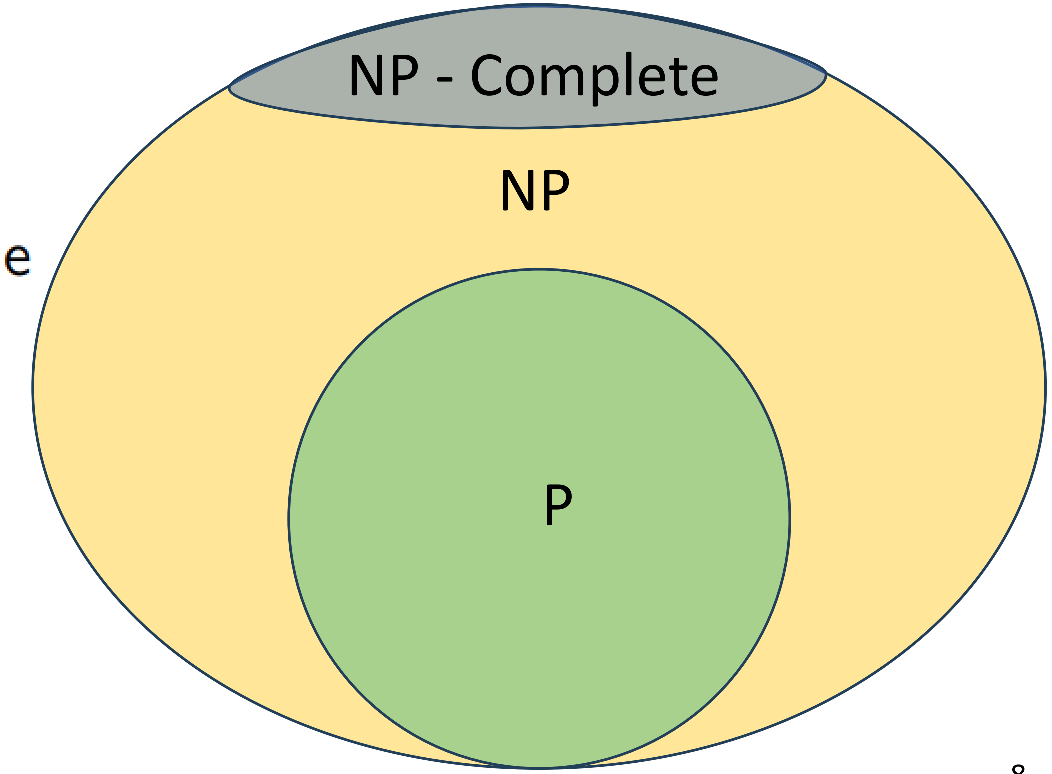
NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\ Y \leq_p X$

$\mathrm{P} \subseteq \mathrm{NP}$ $\qquad$ $\mathrm{NPC} \subseteq \mathrm{NP}$

- Take any $X \in \mathrm{NP}$ and prove that it cannot be solved in poly time
  - You proved $\mathrm{P} \neq \mathrm{NP}$



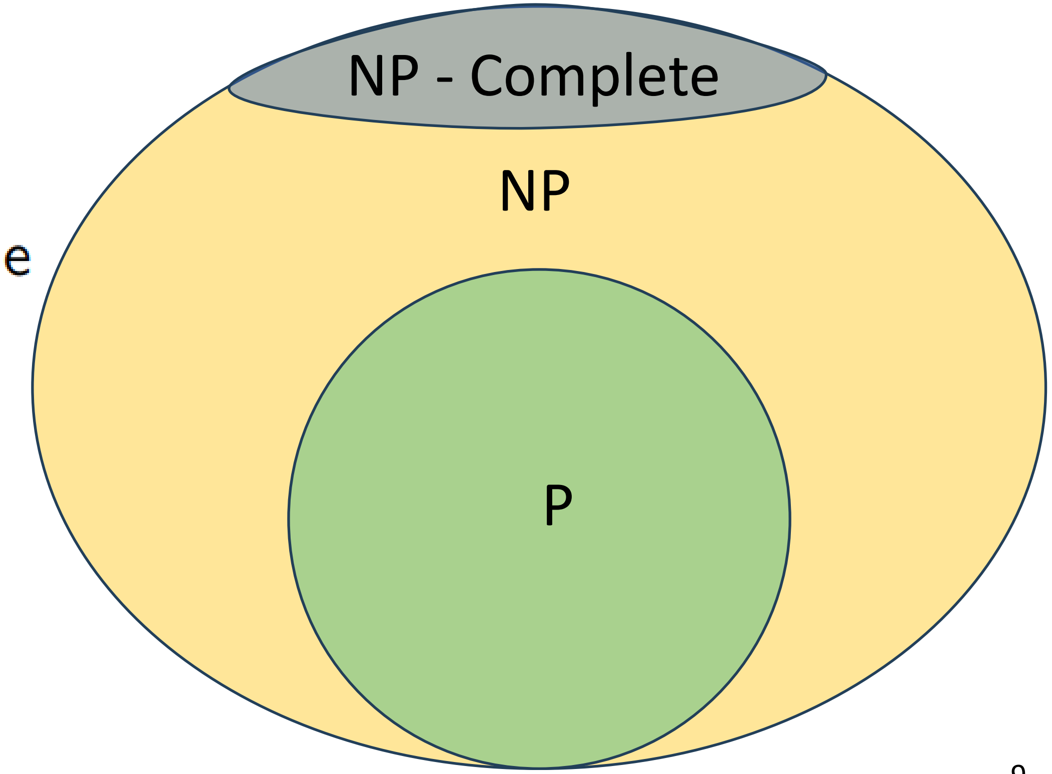
NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\; Y \leq_p X$

$\mathrm{P} \subseteq \mathrm{NP}$ $\quad\quad$ $\mathrm{NPC} \subseteq \mathrm{NP}$

- Take any $X \in \mathrm{NP}$ and prove that it cannot be solved in poly time
  - You proved $\mathrm{P} \neq \mathrm{NP}$
- Take any $X \in \mathrm{NPC}$ and solve it in poly time
  - You proved $\mathrm{P} = \mathrm{NP}$



NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ \ Y \leq_p X$

Why should you prove a problem to be NP-$\textsc{Complete}$?

NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\ Y \leq_p X$

Why should you prove a problem to be NP-COMPLETE?

- Good evidence that it is hard
- Unless your interest is proving $\mathrm{P} = \mathrm{NP}$ stop trying finding efficient algorithm

NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\ Y \leq_p X$

Why should you prove a problem to be NP-$\mathrm{C{\small OMPLETE}}$?

- Good evidence that it is hard
- Unless your interest is proving $\mathrm{P} = \mathrm{NP}$ stop trying finding efficient algorithm

What to tell your boss if you fail to find a fast algorithm for a problem?

NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\ Y \leq_p X$

Why should you prove a problem to be NP-COMPLETE?

- Good evidence that it is hard
- Unless your interest is proving $\mathrm{P} = \mathrm{NP}$ stop trying finding efficient algorithm

What to tell your boss if you fail to find a fast algorithm for a problem?

1. I am too dumb! ▷ You are fired

NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\ Y \leq_p X$

Why should you prove a problem to be NP-COMPLETE?

- Good evidence that it is hard
- Unless your interest is proving $\mathrm{P} = \mathrm{NP}$ stop trying finding efficient algorithm

What to tell your boss if you fail to find a fast algorithm for a problem?

1. I am too dumb! ▷ You are fired
2. There is no fast algorithm! *You claim that* $\mathrm{P} \neq \mathrm{NP}$ ▷ Need a proof

NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\; Y \leq_p X$

Why should you prove a problem to be NP-COMPLETE?

- Good evidence that it is hard
- Unless your interest is proving $\mathrm{P} = \mathrm{NP}$ stop trying finding efficient algorithm

What to tell your boss if you fail to find a fast algorithm for a problem?

1. I am too dumb! ▷ You are fired
2. There is no fast algorithm! *You claim that* $\mathrm{P} \neq \mathrm{NP}$ ▷ Need a proof
3. I cannot solve it, but neither can anyone in the world! ▷ Need reduction

NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ \ Y \leq_p X$

What to tell your boss if you fail to find a fast algorithm for a problem?

Dealing with Hard Problems

- What to do when we find a problem that looks hard...

I couldn't find a polynomial-time algorithm;
I guess I'm too dumb.

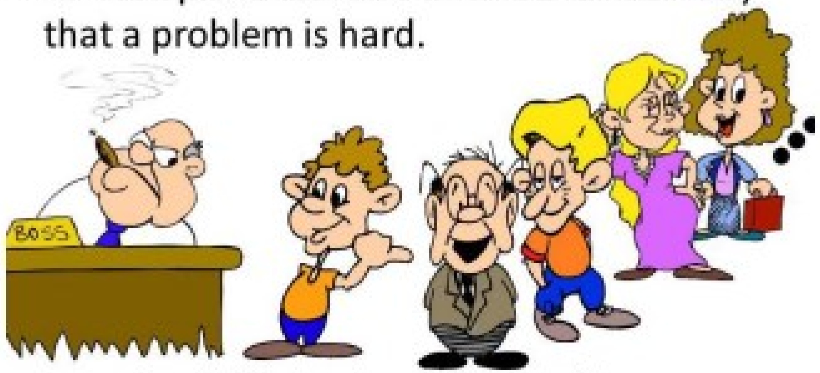
Dealing with Hard Problems

- Sometimes we can prove a strong lower bound... (but not usually)

I couldn't find a polynomial-time algorithm,
because no such algorithm exists!

Dealing with Hard Problems

- NP-completeness let's us show collectively that a problem is hard.



I couldn't find a polynomial-time algorithm,
but neither could all these other smart people.

Proving NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\ \ Y \leq_p X$

How to prove a problem NP-$\textsc{Complete}$?

Proving NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

How to prove a problem NP-Complete?

Can we do so many reductions?

Proving NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\ Y \leq_p X$

To prove X $\textsc{NP-Complete}$, reduce an $\textsc{NP-Complete}$ problem Z to X

Proving NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\, Y \in \mathrm{NP}\ Y \leq_p X$

To prove X $\mathrm{NP\text{-}C{\small OMPLETE}}$, reduce an $\mathrm{NP\text{-}C{\small OMPLETE}}$ problem Z to X

If Z is $\mathrm{NP\text{-}C{\small OMPLETE}}$, and

1. $X \in \mathrm{NP}$
2. $Z \leq_p X$

then X is $\mathrm{NP\text{-}C{\small OMPLETE}}$

Proving NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ \ Y \leq_p X$

To prove X NP-$\textsc{Complete}$, reduce an NP-$\textsc{Complete}$ problem Z to X

Where to begin? we need a first NP-$\textsc{Complete}$ Problem

Proving NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

To prove X $\mathrm{NP\text{-}C{\small OMPLETE}}$, reduce an $\mathrm{NP\text{-}C{\small OMPLETE}}$ problem Z to X

Theorem (The Cook-Levin theorem)

$\textsc{sat}(f)$ *is* $\mathrm{NP\text{-}C{\small OMPLETE}}$

- Proved by Stephen Cook (1971) and earlier by Leonid Levin (but became known later)
- Levin proved six $\mathrm{NP\text{-}C{\small OMPLETE}}$ problems (in addition to other results)

Proving NP-Complete Problems

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

To prove X NP-Complete, reduce an NP-Complete problem Z to X

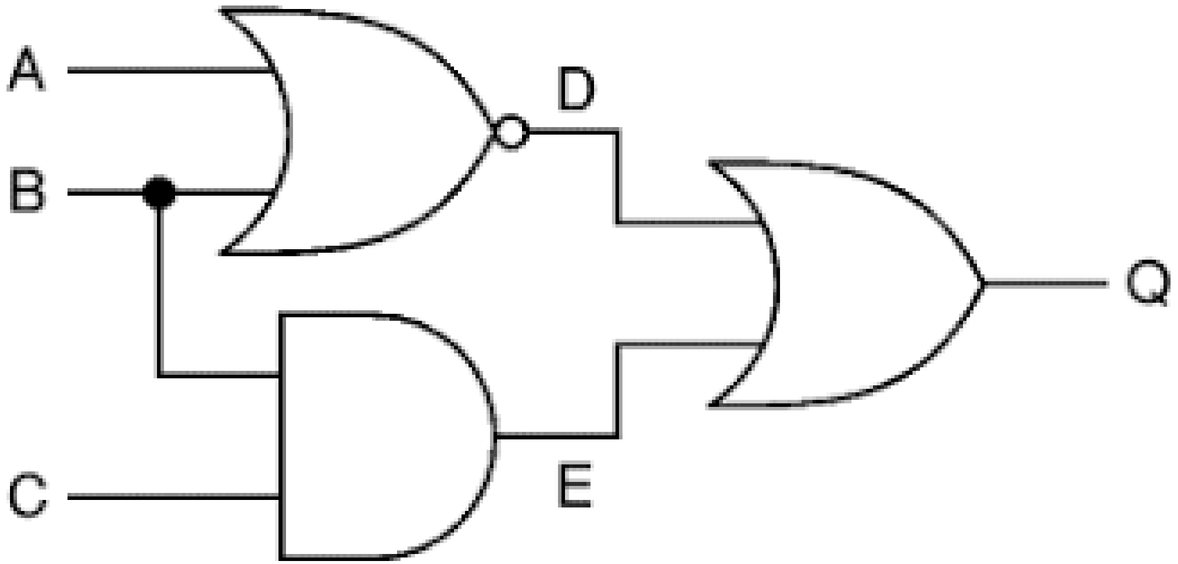
Theorem (The Cook-Levin theorem)

SAT(f) is NP-Complete

CIRCUIT-SAT is NP-Complete

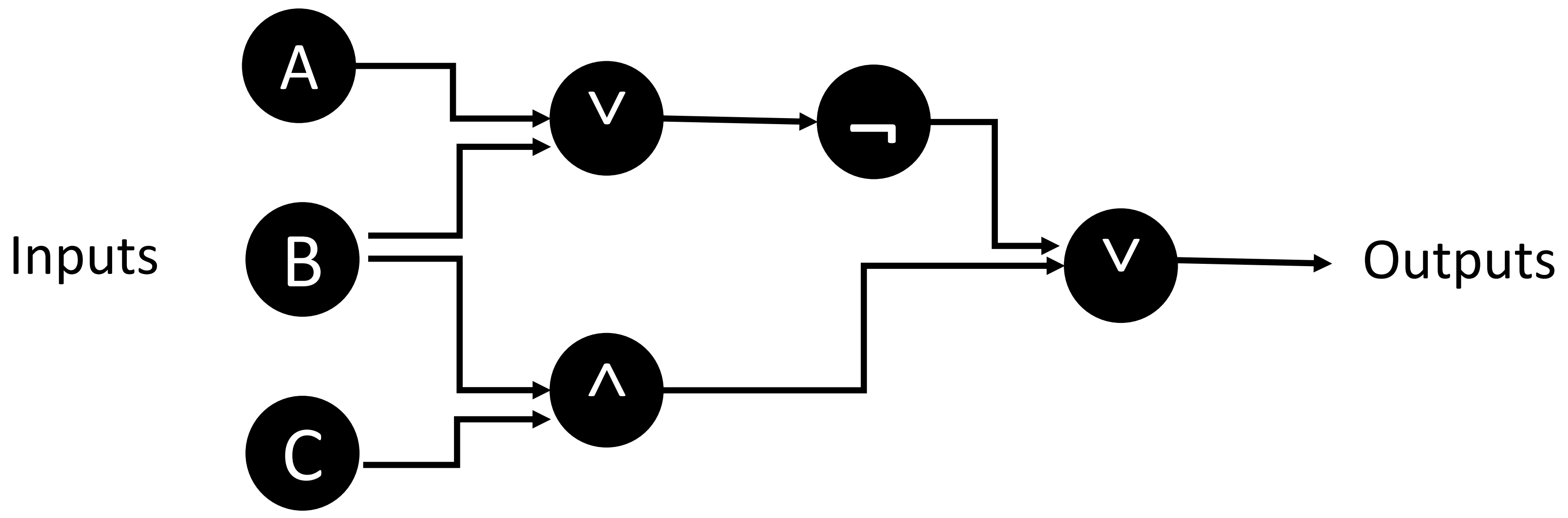
What is Circuit SAT problem?

Circuit-SAT is the decision problem of determining whether a given Boolean circuit has an assignment of its inputs that makes the output ***TRUE***



What is Circuit SAT problem?

Circuit-SAT is the decision problem of determining whether a given Boolean circuit has an assignment of its inputs that makes the output ***TRUE***



Claim: 3-Sat is NP-Complete

If Z is NP-COMPLETE, and

1. $X \in \mathrm{NP}$
2. $Z \leq_p X$

then X is NP-COMPLETE

1) 3-SAT ϵ NP

Claim 3-Sat is NP-Complete

If Z is NP-COMPLETE, and

1. $X \in \mathrm{NP}$
2. $Z \leq_p X$

then X is NP-COMPLETE

1) **3-SAT** є NP

Certificate:

T or ***F*** for each variable

Verifier:

- Check cert has the right format
- Check that formula evaluates to T

Claim 3-Sat is NP-Complete

If Z is NP-COMPLETE, and
1. $X \in \mathrm{NP}$
2. $Z \leq_p X$

then X is NP-COMPLETE

1) **3-SAT** ϵ NP

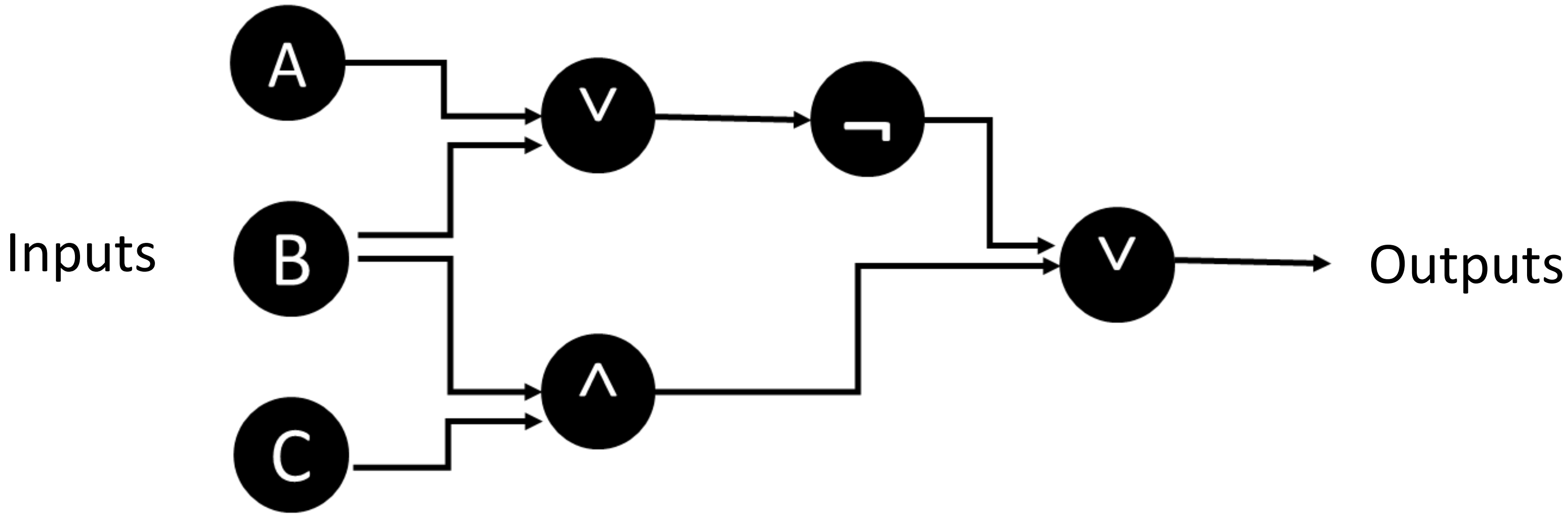
2) **NP-Complete $\leq_p$ 3-SAT**

   Circuit-SAT $\leq_p$ 3-SAT

Claim 3-Sat is NP-Complete

2) NP-Complete $\leq_{\mathbf{p}}$ 3-SAT Circuit-SAT $\leq_{\mathrm{p}}$ 3-SAT

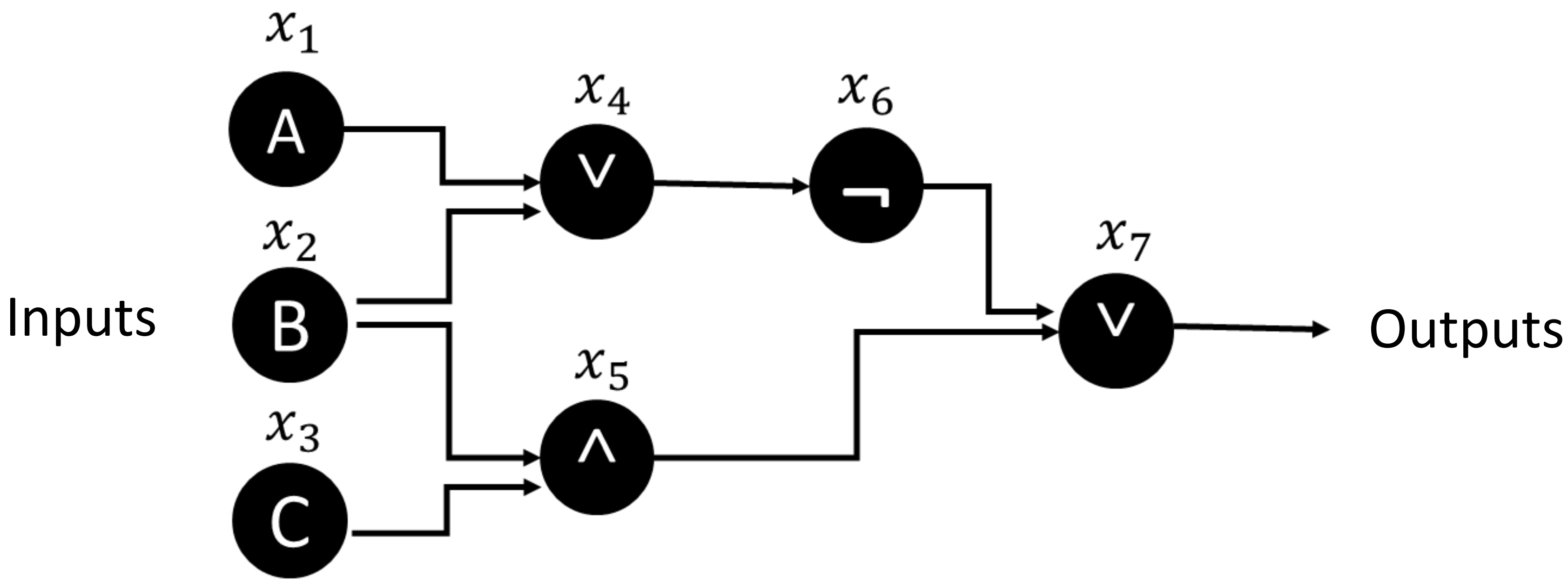
Assign variable for every gate



Claim 3-Sat is NP-Complete

2) NP-Complete $\leq_{\mathbf{p}}$ 3-SAT Circuit-SAT $\leq_{\mathrm{p}}$ 3-SAT

Assign variable for every gate



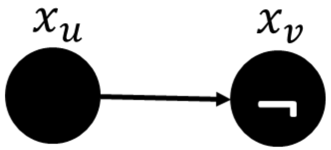
Claim 3-Sat is NP-Complete

2) NP-Complete $\leq_p$ 3-SAT

Circuit-SAT $\leq_p$ 3-SAT

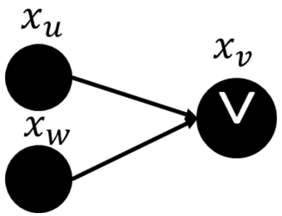
Assign variable for every gate

For each **NOT** gate:



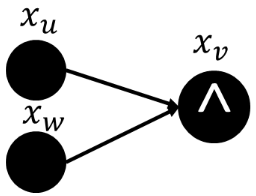
$(x_u \vee x_v) \wedge (\overline{x_u} \vee \overline{x_v})$

For each **OR** gate:

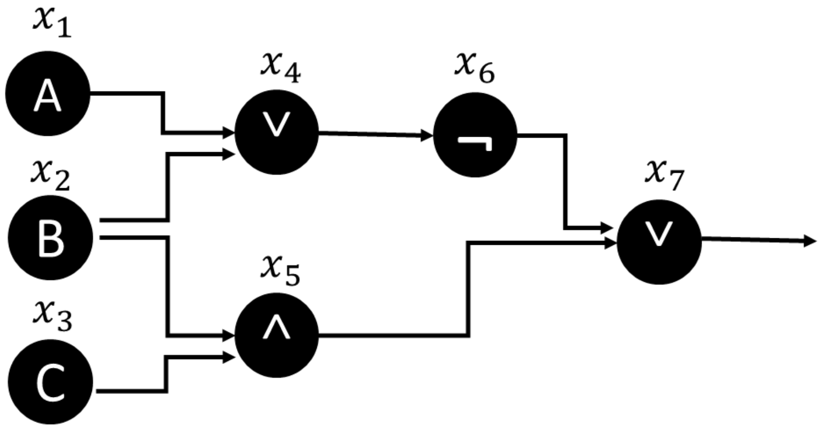


$(x_v \vee \overline{x_u}) \wedge (x_v \vee \overline{x_w}) \wedge (\overline{x_v} \vee x_u \vee x_w)$

For each **AND** gate:



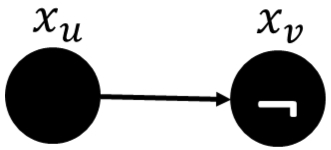
$(\overline{x_v} \vee x_u) \wedge (\overline{x_v} \vee x_w) \wedge (x_v \vee \overline{x_u} \vee \overline{x_w})$



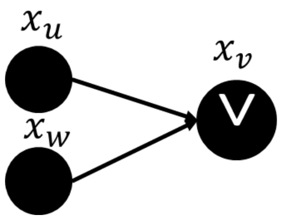
Claim 3-Sat is NP-Complete

2) NP-Complete $\leq_p$ 3-SAT   Circuit-SAT $\leq_p$ 3-SAT

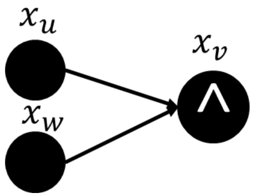
Assign variable for every gate

For each **NOT** gate: 

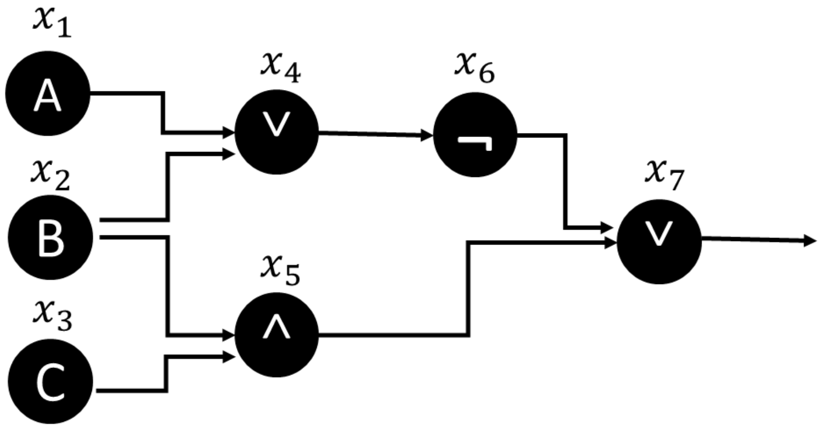
$(x_u \vee x_v) \wedge (\overline{x_u} \vee \overline{x_v})$

For each **OR** gate: 

$(x_v \vee \overline{x_u}) \wedge (x_v \vee \overline{x_w}) \wedge (\overline{x_v} \vee x_u \vee x_w)$

For each **AND** gate: 

$(\overline{x_v} \vee x_u) \wedge (\overline{x_v} \vee x_w) \wedge (x_v \vee \overline{x_u} \vee \overline{x_w})$



AND together clauses for every gate

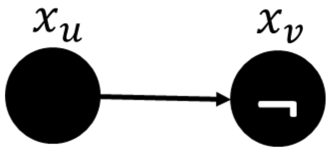
Claim 3-Sat is NP-Complete

2) NP-Complete $\leq_p$ 3-SAT

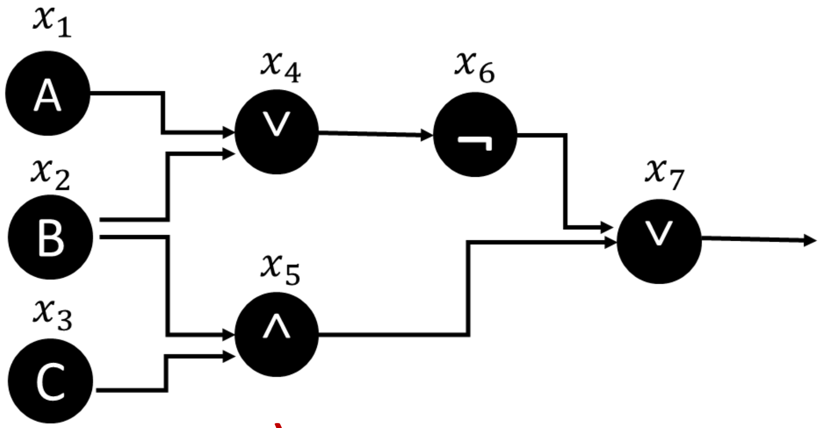
Circuit-SAT $\leq_p$ 3-SAT

Assign variable for every gate

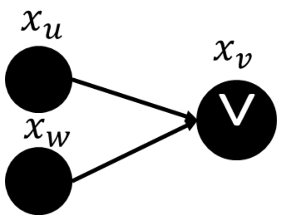
For each **NOT** gate:



$(x_u \vee x_v \vee \text{FALSE}) \wedge (\overline{x_u} \vee \overline{x_v} \vee \text{FALSE})$

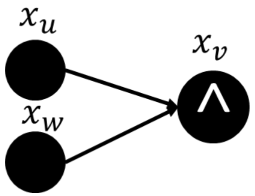


For each **OR** gate:



$(x_v \vee \overline{x_u} \vee \text{FALSE}) \wedge (x_v \vee \overline{x_w} \vee \text{FALSE}) \wedge (\overline{x_v} \vee x_u \vee x_w)$

For each **AND** gate:



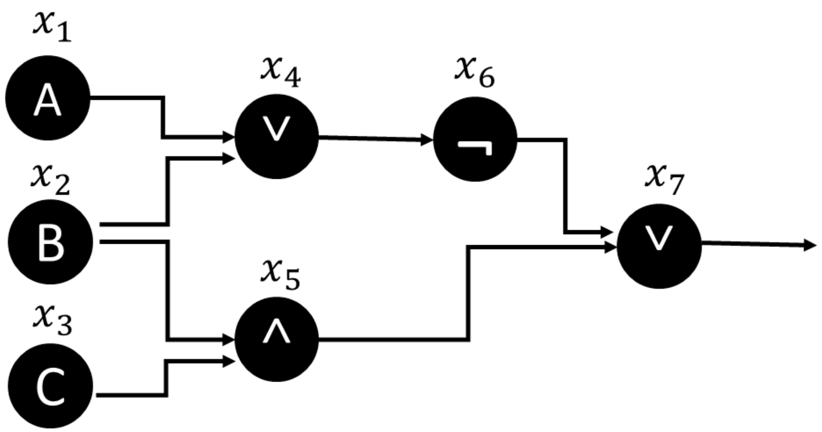
AND together clauses for every gate

$(\overline{x_v} \vee x_u \vee \text{FALSE}) \wedge (\overline{x_v} \vee x_w \vee \text{FALSE}) \wedge (x_v \vee \overline{x_u} \vee \overline{x_w})$

Claim 3-Sat is NP-Complete

2) NP-Complete $\leq_p$ 3-SAT

Circuit-SAT $\leq_p$ 3-SAT



$$(x_4 \vee x_6 \vee \text{FALSE}) \wedge (\overline{x_4} \vee \overline{x_6} \vee \text{FALSE})$$

$$\wedge$$

$$(x_4 \vee \overline{x_1} \vee \text{FALSE}) \wedge (x_4 \vee \overline{x_2} \vee \text{FALSE}) \wedge (\overline{x_4} \vee x_1 \vee x_2)$$

$$\wedge$$

$$(\overline{x_5} \vee x_2 \vee \text{FALSE}) \wedge (\overline{x_5} \vee 3 \vee \text{FALSE}) \wedge (x_5 \vee \overline{x_2} \vee \overline{x_3})$$

$$\wedge$$

$$(x_7 \vee \overline{x_6} \vee \text{FALSE}) \wedge (x_7 \vee \overline{x_5} \vee \text{FALSE}) \wedge (\overline{x_7} \vee x_6 \vee x_5)$$

Thanks a lot

“In case I don’t see you again,

Good Morning, **Good Afternoon** and **Good Evening**” – Jim Carry