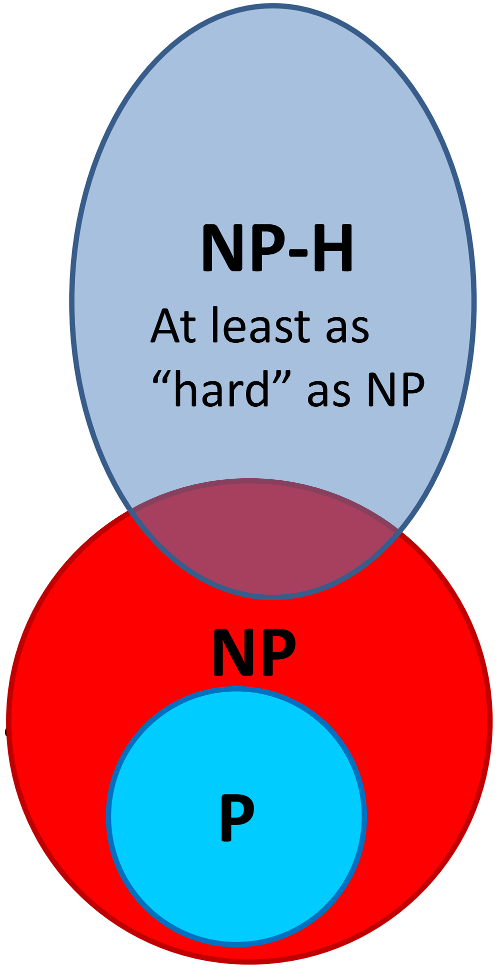
CS3102 Theory of Computation

Problem Types

- Decision Problems:
  - Is there a solution?
    - Output is True/False
  - **Can** all these boxes fit in the trunk of my car?
- Search Problems:
  - Find a solution
    - Output is complex
  - Show me how to make these boxes fit in the trunk of my car.
- Verification Problems:
  - Given a potential solution, is it valid?
    - Output is True/False
  - Will the boxes fit in the trunk of your care if you load them like this?

NP-Hard

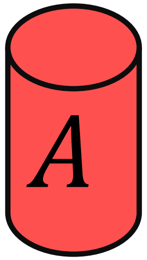
- How can we try to figure out if P=NP?
- Identify problems at least as “hard” as NP
  - If any of these “hard” problems can be solved in polynomial time, then all NP problems can be solved in polynomial time.
- Definition: NP-Hard:
  - B is NP-Hard if $\forall A \in NP, A \leq_p B$
  - $A \leq_p B$ means A reduces to B in polynomial time



MacGyver's Reduction

Problem known to be "hard"

Opening a door



Solution for $\boldsymbol{A}$

Keg cannon battering ram

Aim duct at door, insert keg

Put fire under the Keg

Reduction

Problem of uknown "hardness"

Lighting a fire

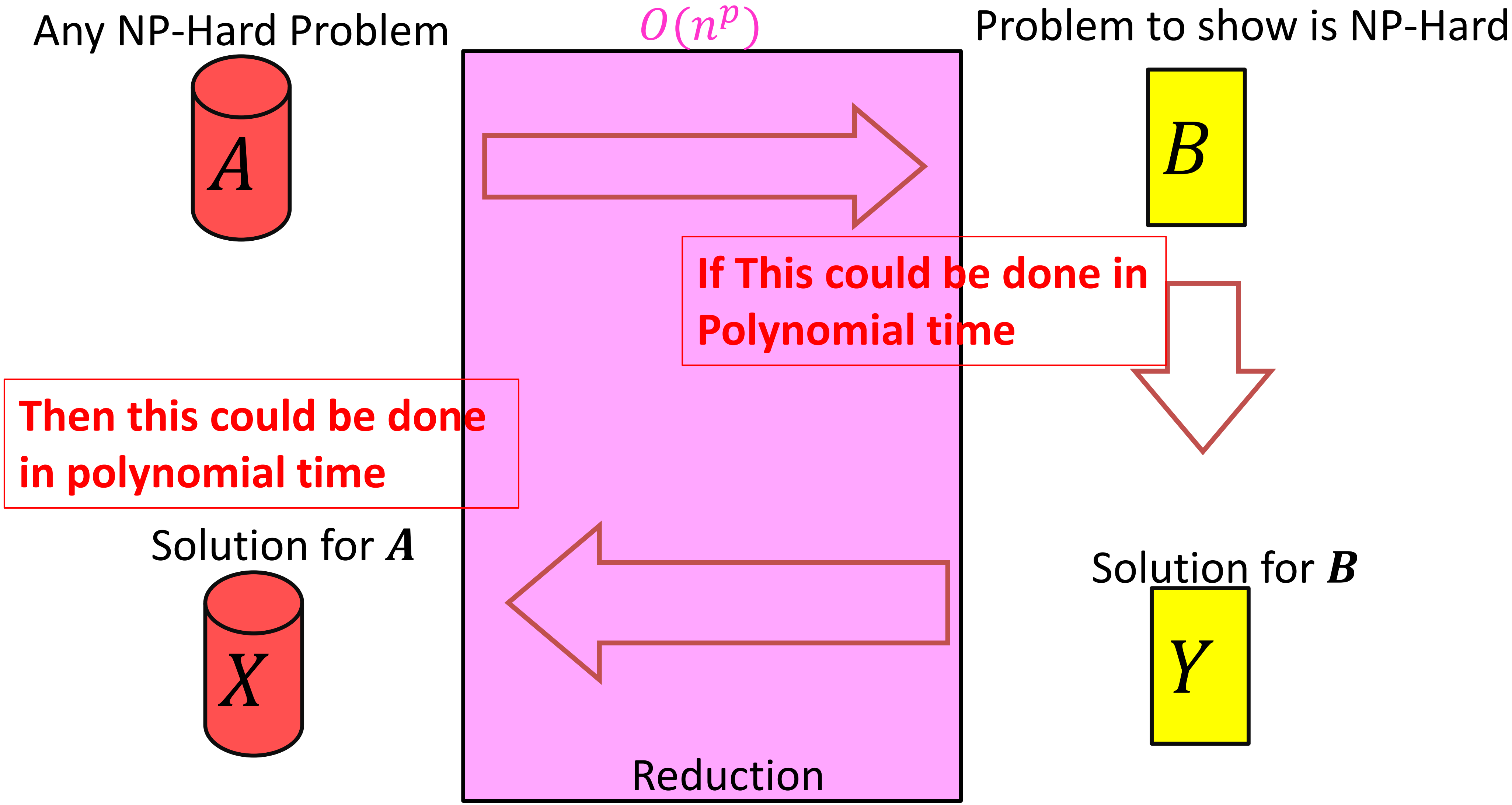
B

How?

Solution for $\boldsymbol{B}$

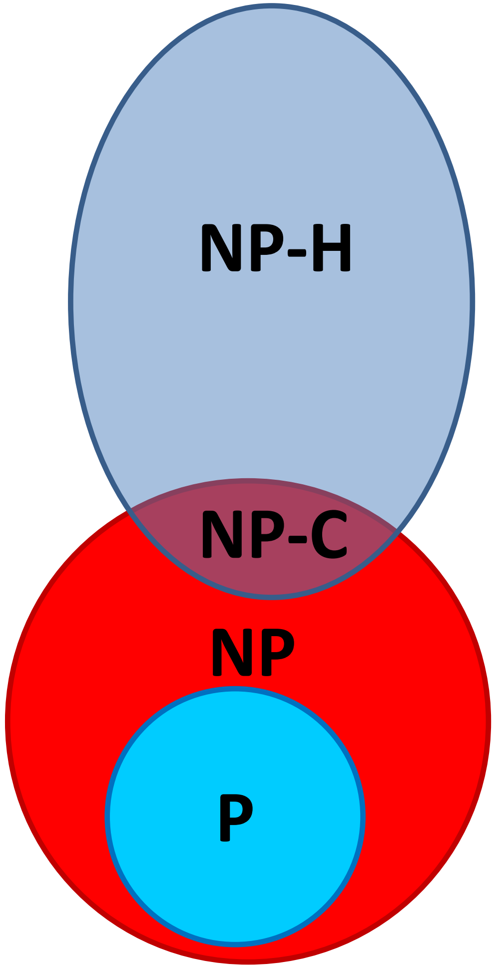
Alcohol, wood, matches

NP-Hardness Reduction



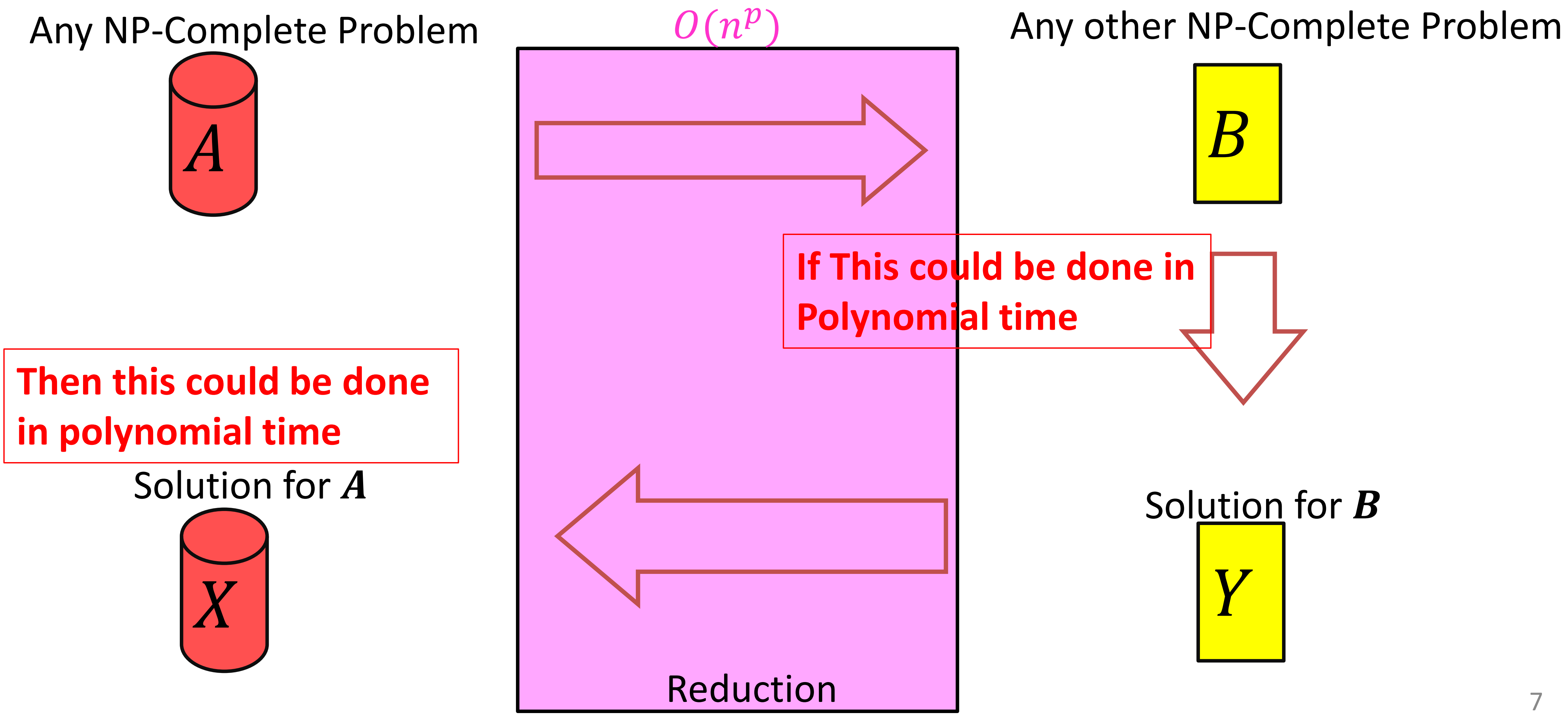
NP-Complete

- "Together they stand, together they fall"
- Problems solvable in polynomial time iff ALL NP problems are
- NP-Complete = NP ∩ NP-Hard
- How to show a problem is NP-Complete?
  - Show it belongs to NP
    - Give a polynomial time verifier
  - Show it is NP-Hard
    - Give a reduction from another NP-H problem

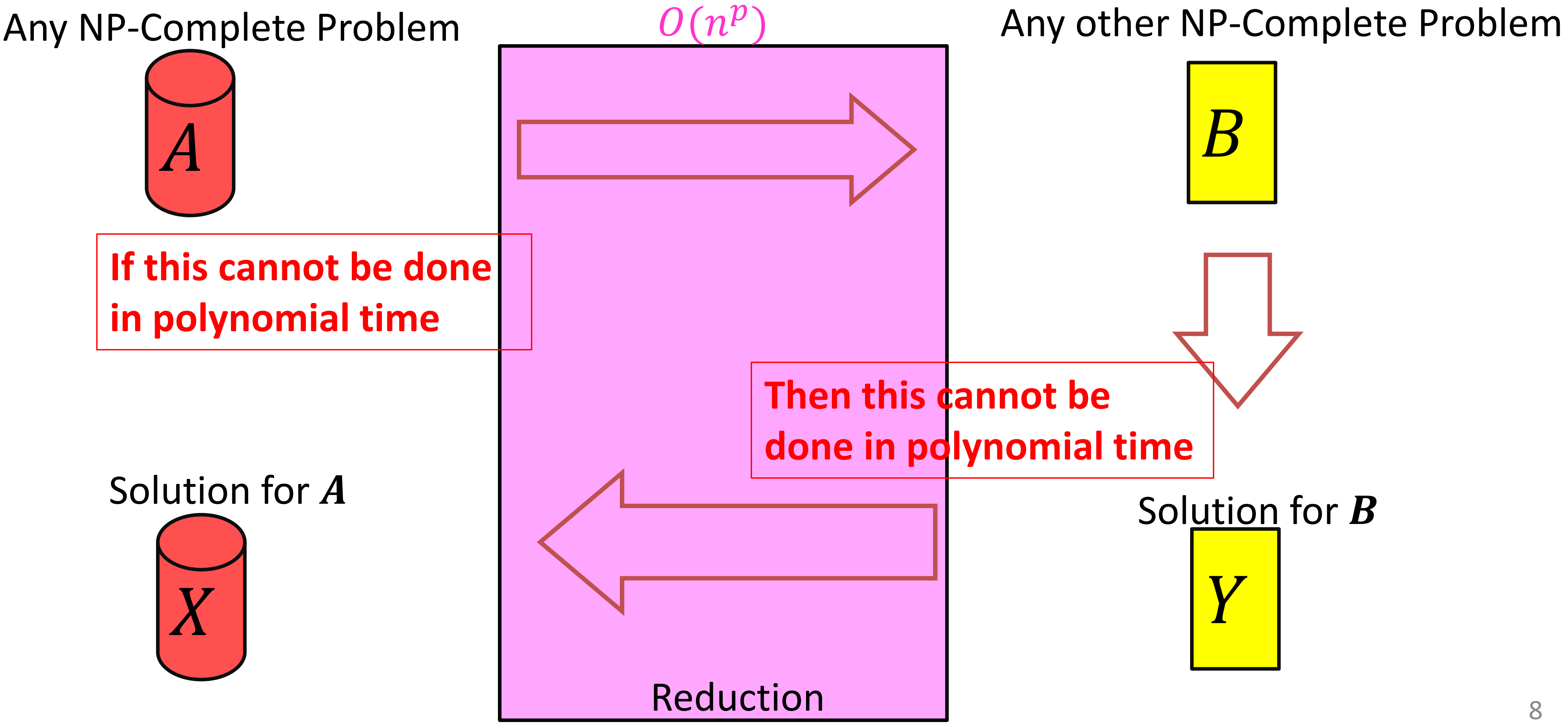


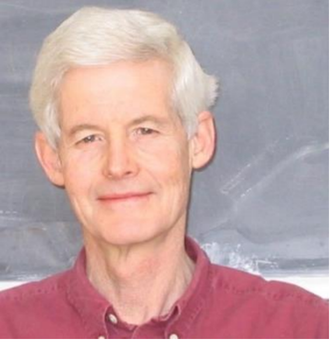
We now just need a FIRST NP-Hard problem

NP-Completeness



NP-Completeness





Stephen Cook

Leonid Levin

3-SAT

- Shown to be NP-Hard by Cook and Levin (independently)
- Given a 3-CNF formula (logical AND of clauses, each an OR of 3 variables), Is there an assignment of true/false to each variable to make the formula true?

$$\underbrace{(x \vee y \vee z)}_{\text{Clause}} \wedge (x \vee \bar{y} \vee y) \wedge (u \vee y \vee \bar{z}) \wedge (z \vee \bar{x} \vee u) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$$

Variables

$$\boldsymbol{x = true}$$
$$\boldsymbol{y = false}$$
$$\boldsymbol{z = false}$$
$$\boldsymbol{u = true}$$

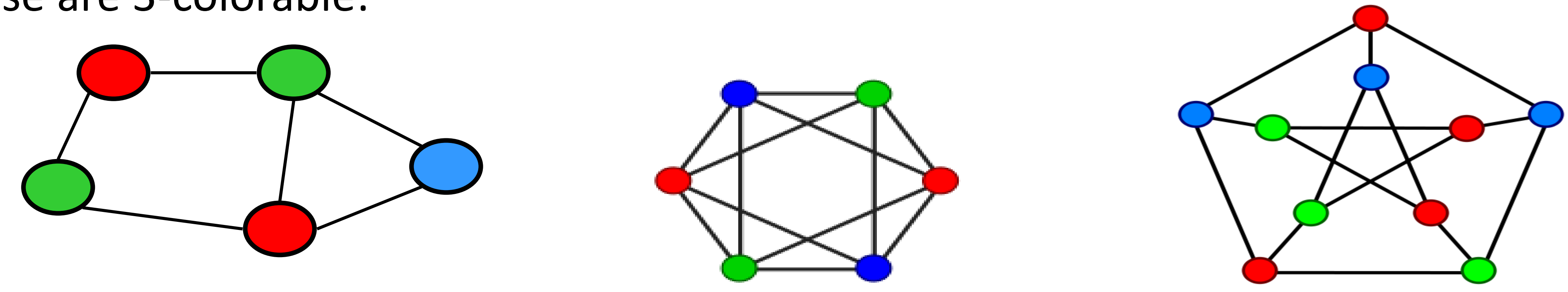
Proof idea

- For a given non-determinstic polynomial time TM and an input string:
  - Create variables representing configurations of the TM
  - Create Clauses to represent valid transitions among configurations
  - Formula will be satisfiable if and only if the machine accepts the input
- Conclusion: If we can decide 3SAT in polynomial time, we can simulate any non-deterministic polynomial time TM in polynomial time

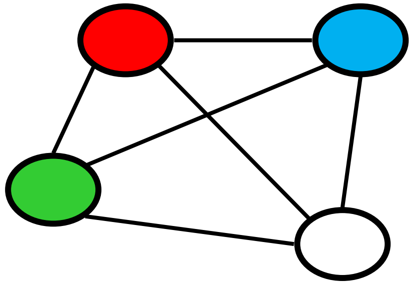
Another NP-Complete Problem

Graph 3-coloring: given a graph, is it 3-colorable? (adjacent nodes get different colors)

These are 3-colorable:



This is not 3-colorable:



How do we know $3col \in NP$?

Graph Colorability

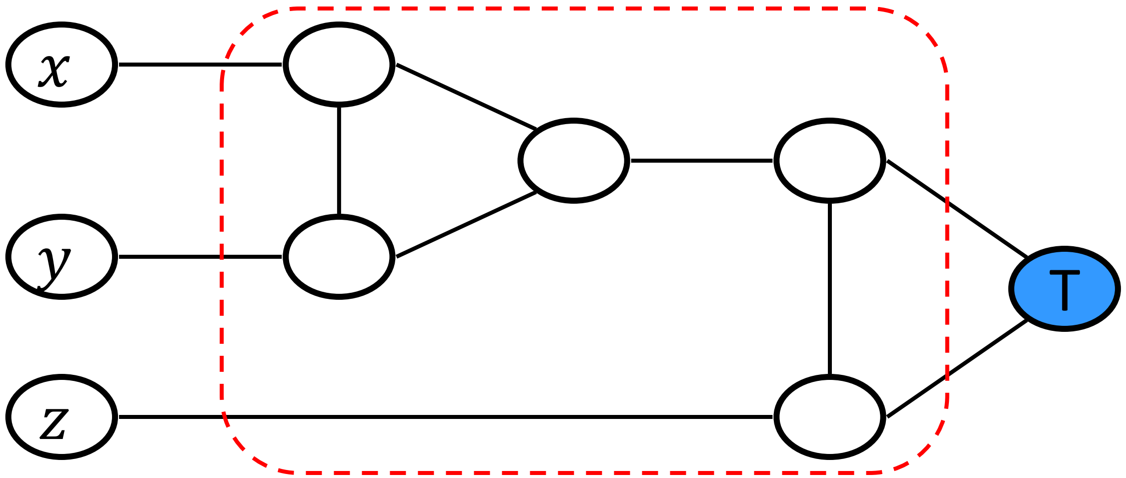
Problem: is a given graph G 3-colorable?

Theorem: Graph 3-colorability is NP-complete.

Proof: Reduction from 3-SAT.

Idea: construct a colorability "OR gate" "gadget":

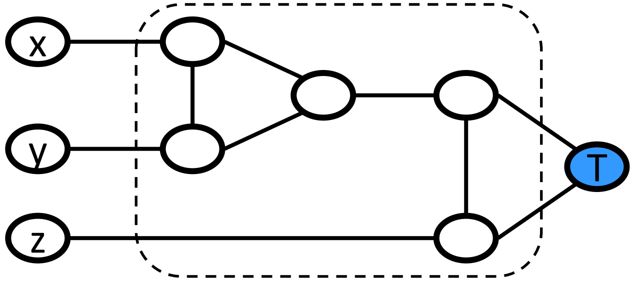
$(x \vee y \vee z) \quad \Rightarrow$

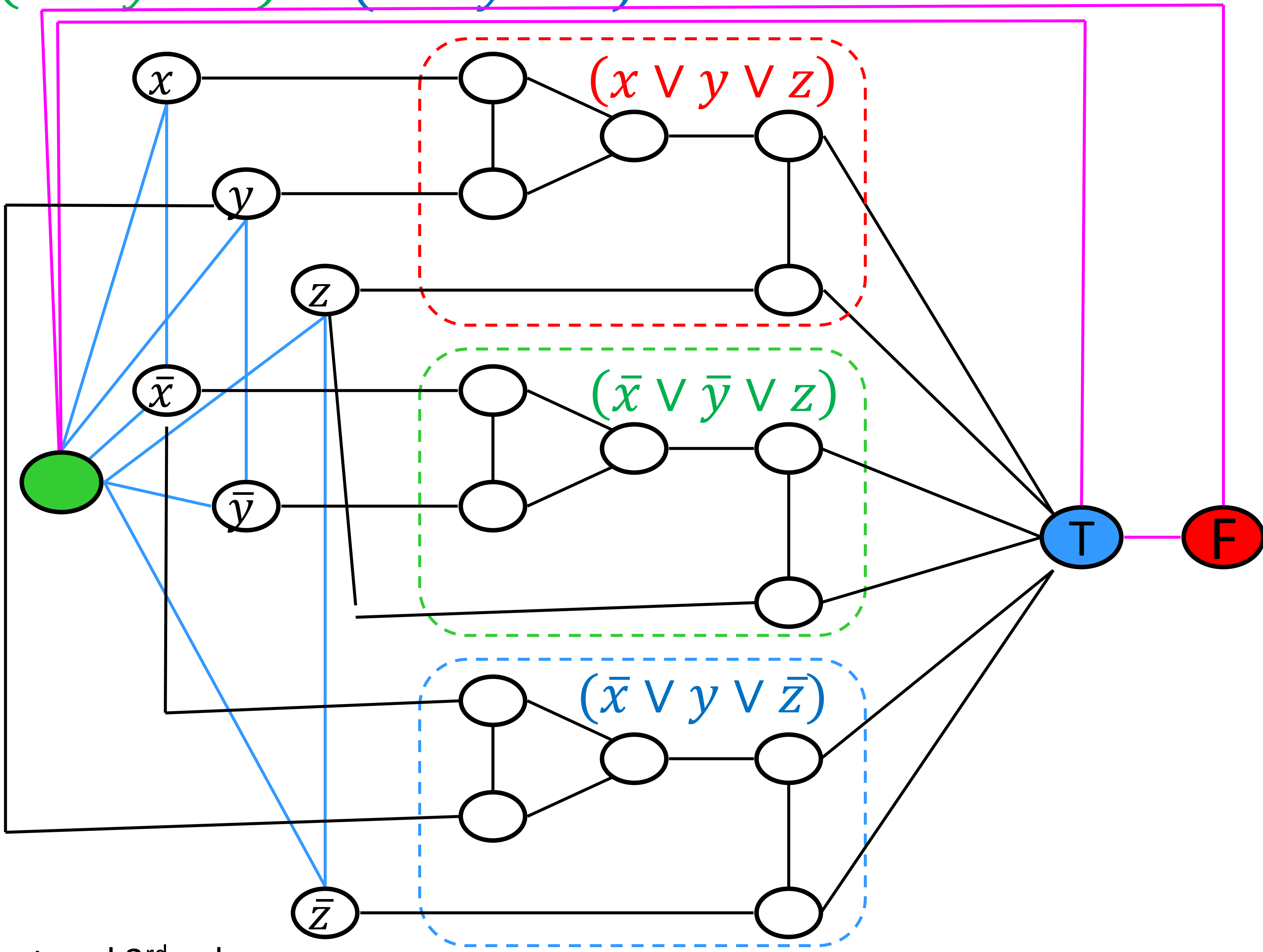


Property: gadget is 3-colorable iff $(x + y + z)$ is true

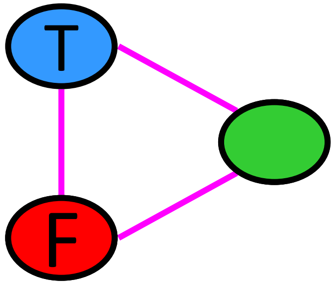
Example: $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z})$

"Or Gate":

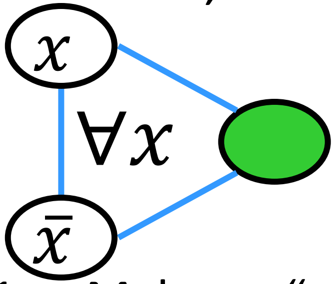




Makes T/F different colors:



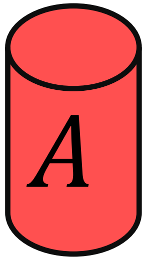
Makes $x, \bar{x}$ different colors:



1. Make an "or gate" for each clause
2. Add a node for False and 3rd color
3. Connect T,F,3rd into a triangle
4. Connect each node to its complement and 3rd color

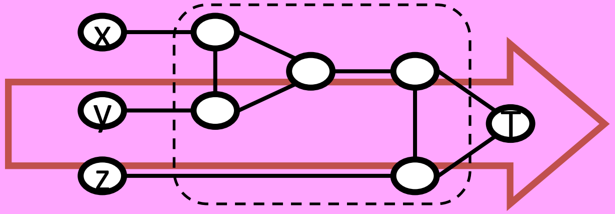
NP-Completeness

3-SAT

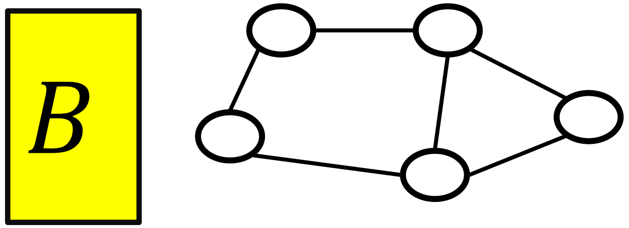


$(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z})$

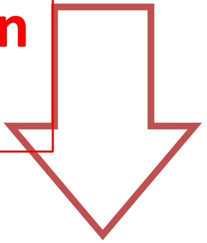
$O(n^p)$



3-Colorability

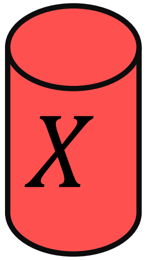


If This could be done in Polynomial time

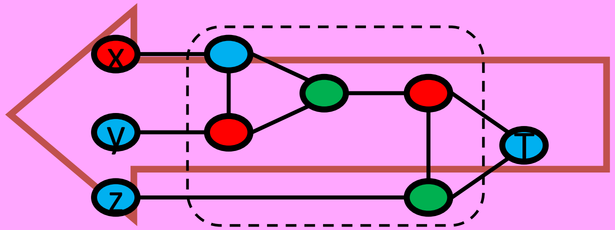


Then this could be done in polynomial time

Solution for ***A***

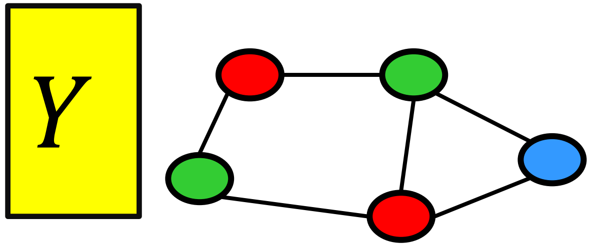


$x = False$
$y = True$
$z = True$



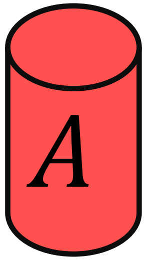
Reduction

Solution for ***B***



NP-Completeness

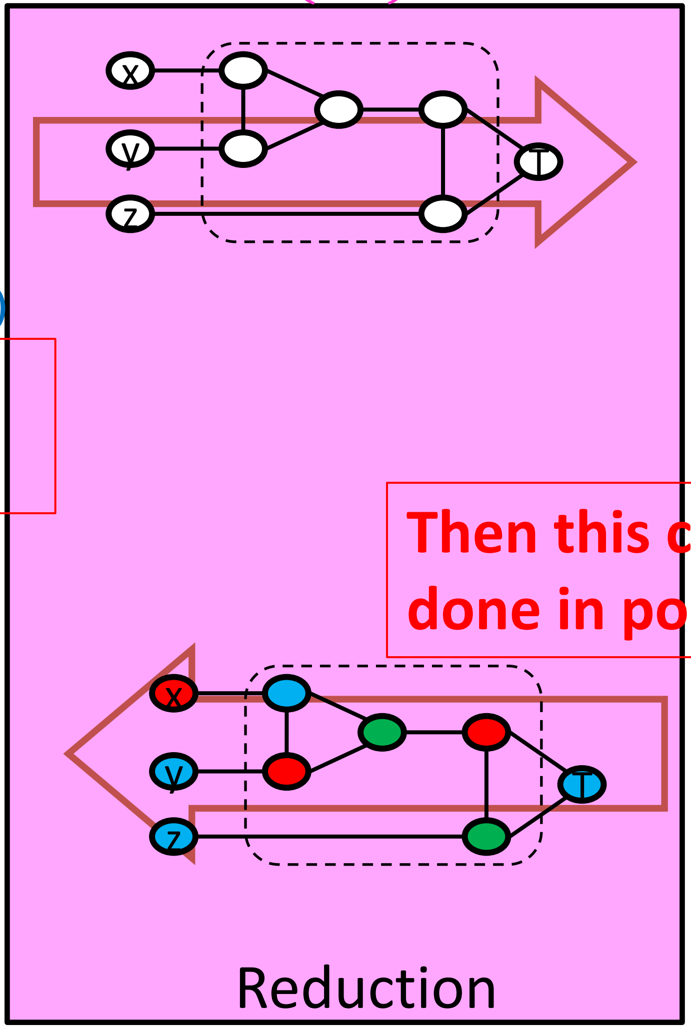
3-SAT



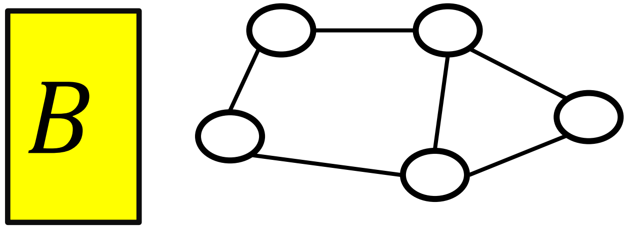
$(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z})$

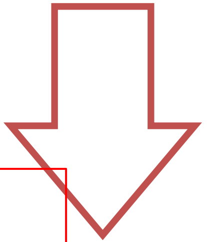
If this cannot be done in polynomial time

$O(n^p)$



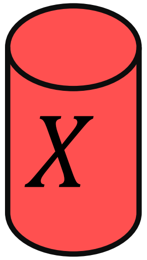
3-Colorability





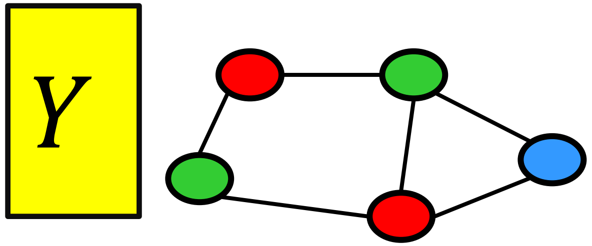
Then this cannot be done in polynomial time

Solution for $\boldsymbol{A}$



$x = False$
$y = True$
$z = True$

Solution for $\boldsymbol{B}$



What about Search Problems

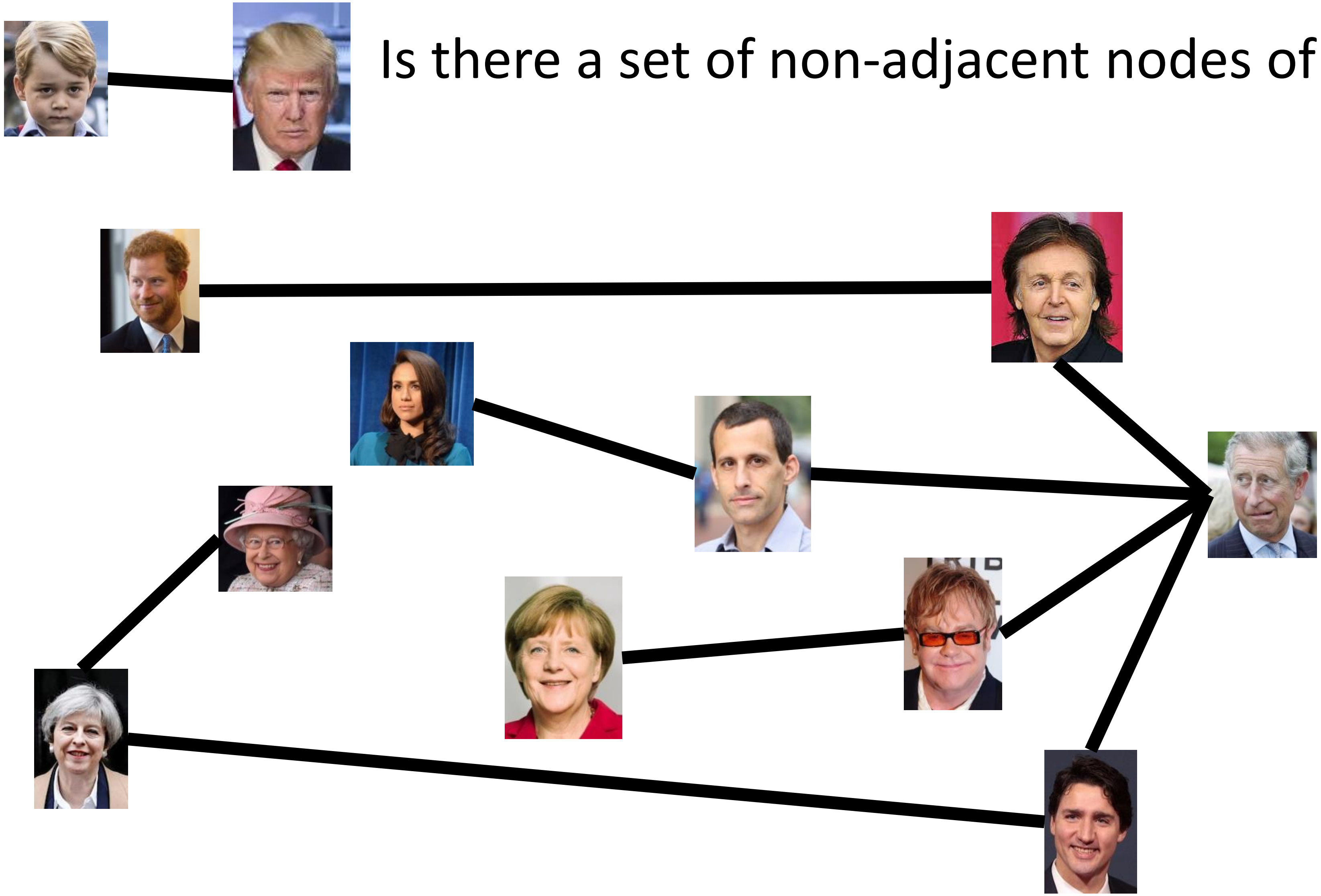
- If we can solve a decision version in polynomial time, we can solve the search version as well.
- Idea: use the decider to build a solution "guess and check" one piece at a time

Search-Decision Reduction

- Given a 3-SAT decider, create a 3-SAT Solver.
- To find assignment for:
  - $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z})$
- Ask decider if this formula is satisfiable:
  - $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee x \vee x)$
  - This is satisfiable if and only if there exists a satisfying assignment where $x = True$
- If yes, ask decider if this is satisfiable:
  - $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee x \vee x) \wedge (y \vee y \vee y)$
- If no, ask decider if this is satisfiable
  - $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{x} \vee \bar{x}) \wedge (y \vee y \vee y)$
- Repeat until you have an assignment for all variables

k Independent Set

Is there a set of non-adjacent nodes of size k?



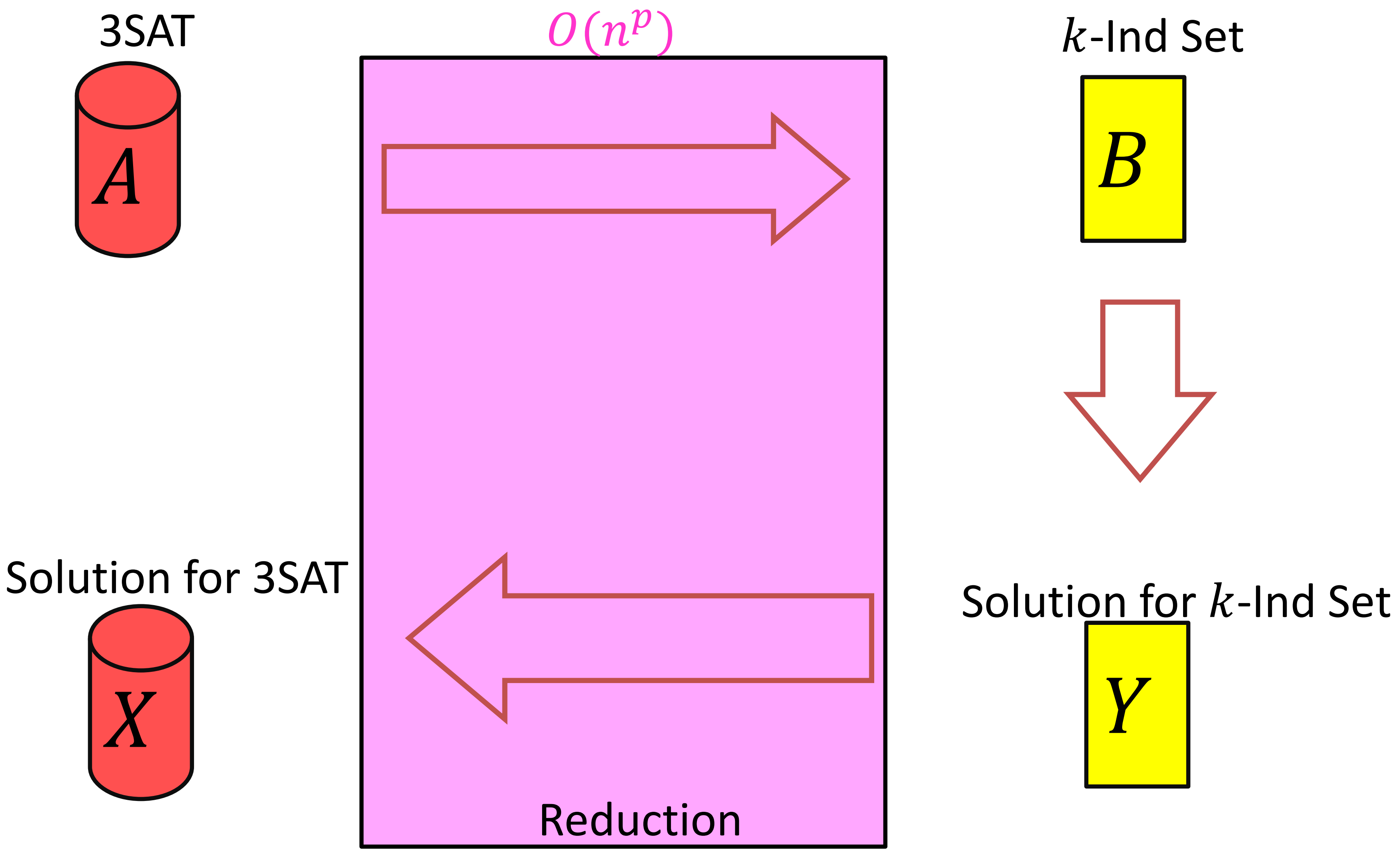
k-Independent Set is NP

- To show: Given a potential solution, can we verify it in $O(n^p)$? $[n = V + E]$

How can we verify it?

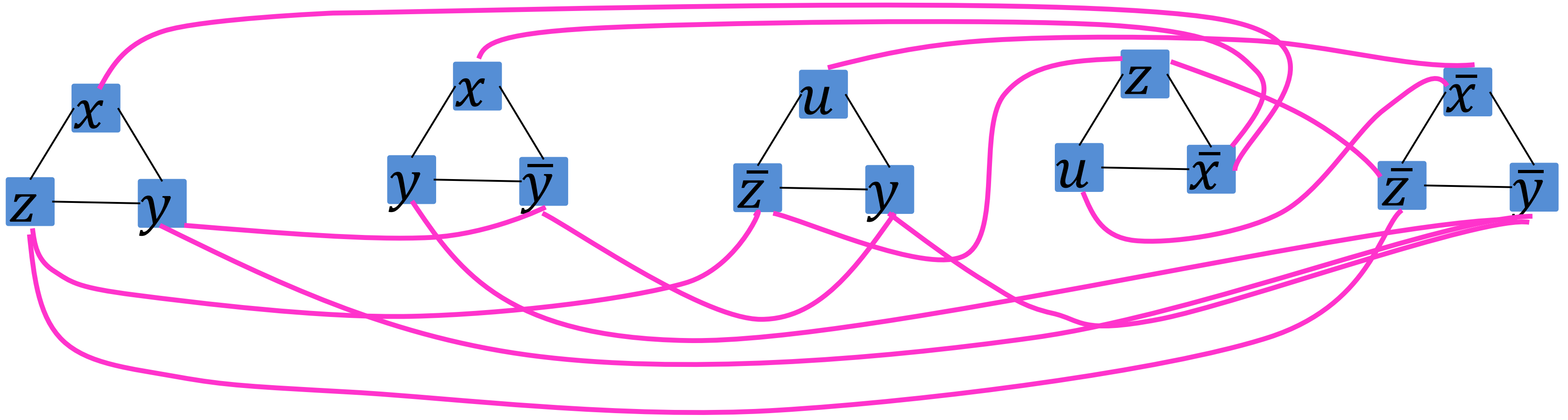
1. Check that it's of size k $O(V)$
2. Check that it's an independent set $O(V^2)$

$3SAT \leq_p kIndSet$



Instance of 3SAT to Instance of kIndSet

$$(x \vee y \vee z) \wedge (x \vee \bar{y} \vee y) \wedge (u \vee y \vee \bar{z}) \wedge (z \vee \bar{x} \vee u) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$$



For each clause, produce a triangle graph with its three variables as nodes

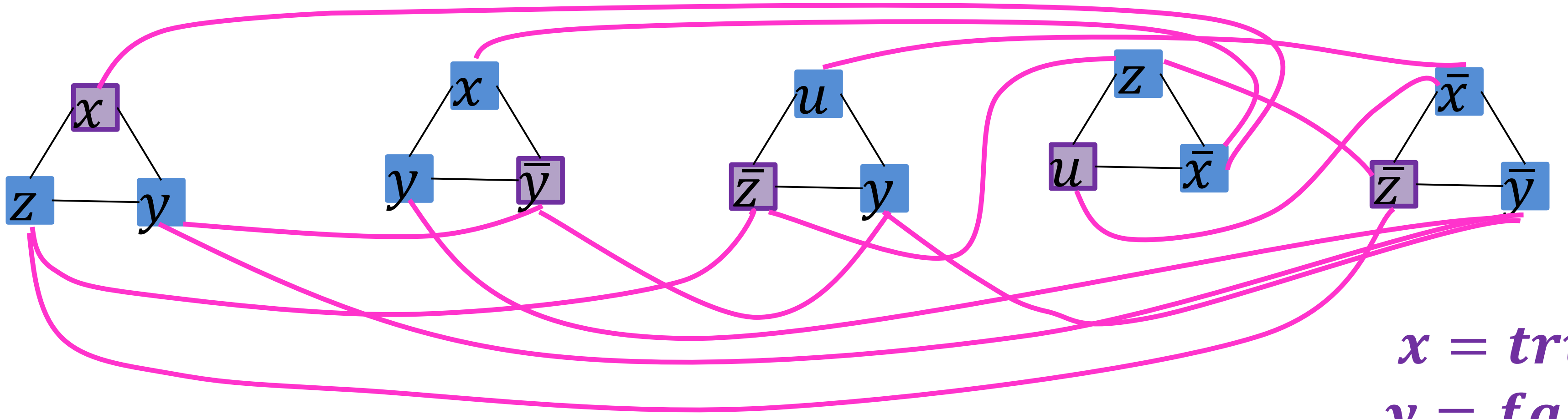
Connect each node to all of its opposites

Let $k =$ number of clauses

There is a k-IndSet in this graph, iff there is a satisfying assignment

kIndSet $\Rightarrow$ Satisfying Assignment

$(x \vee y \vee z) \wedge (x \vee \bar{y} \vee y) \wedge (u \vee y \vee \bar{z}) \wedge (z \vee \bar{x} \vee u) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$



$\boldsymbol{x = true}$
$\boldsymbol{y = false}$
$\boldsymbol{z = false}$
$\boldsymbol{u = true}$

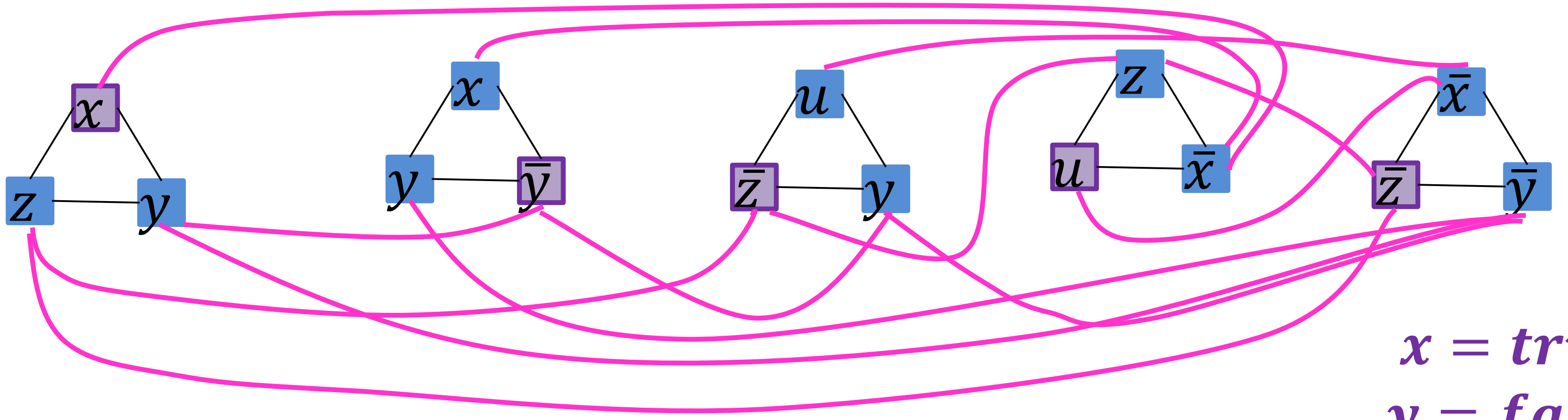
One node per triangle is in the Independent set:
because we can have exactly k total in the set,
and 2 in a triangle would be adjacent

If x is selected in some triangle, $\bar{x}$ is not selected in any triangle:
Because every x is adjacent to every $\bar{x}$

Set the variable which each included node represents to “true”

Satisfying Assignment $\Rightarrow k$IndSet

$$(x \vee y \vee z) \wedge (x \vee \bar{y} \vee y) \wedge (u \vee y \vee \bar{z}) \wedge (z \vee \bar{x} \vee u) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$$



$\boldsymbol{x = true}$
$\boldsymbol{y = false}$
$\boldsymbol{z = false}$
$\boldsymbol{u = true}$

Use one true variable from the assignment for each triangle

The independent set has k nodes, because there are k clauses

If any variable x is true then $\bar{x}$ cannot be true

$3SAT \leq_p kIndSet$