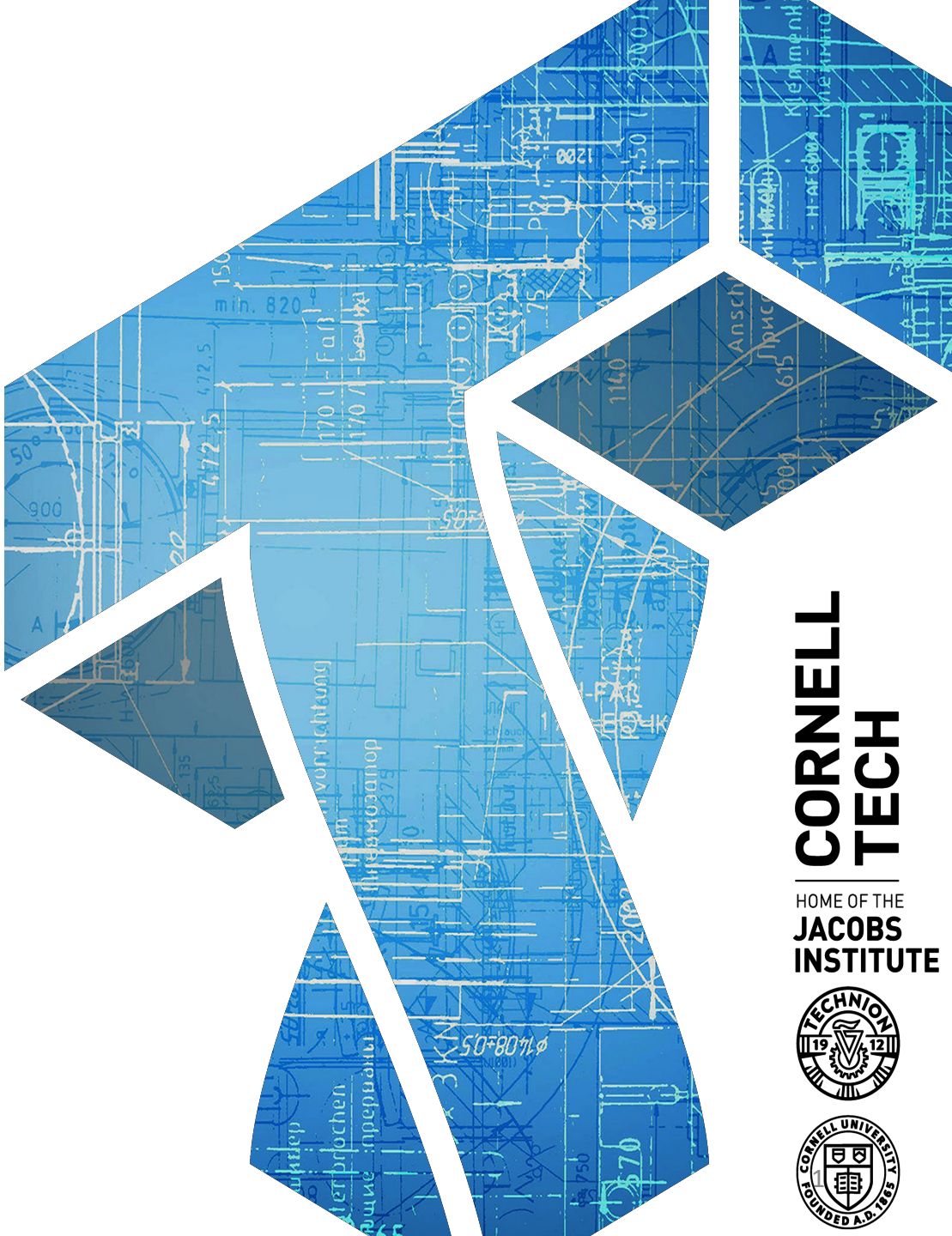
CS 5112 – 11/2
NP-completeness



CORNELL TECH

HOME OF THE JACOBS INSTITUTE

Different versions of the independent set problem

- **Search problem:** Given a graph G = (V,E), find independent set of maximum size

- **Optimization (max) problem:** Given a graph G = (V,E), what is the largest independent set?

- **Decision problem:** Given a graph G = (V,E) and a number k, does G contain an independent set of size k?

Reductions

Def. Problem X reduces to problem Y if arbitrary instances of problem X can be solved using:

1. Polynomial number of standard computational steps, plus
2. Polynomial number of calls to subroutine (oracle) that solves Y

Notation: we write this as X $\leq_P$ Y

Transitivity: X $\leq_P$ Y and Y $\leq_P$ Z implies X $\leq_P$ Z

Polytime equivalence: X $\leq_P$ Y and Y $\leq_P$ X means X, Y equally hard to solve

Ind-Set-Search $\overset{?}{\leq_P}$ Ind-Set-Max $\overset{?}{\leq_P}$ Ind-Set-Dec

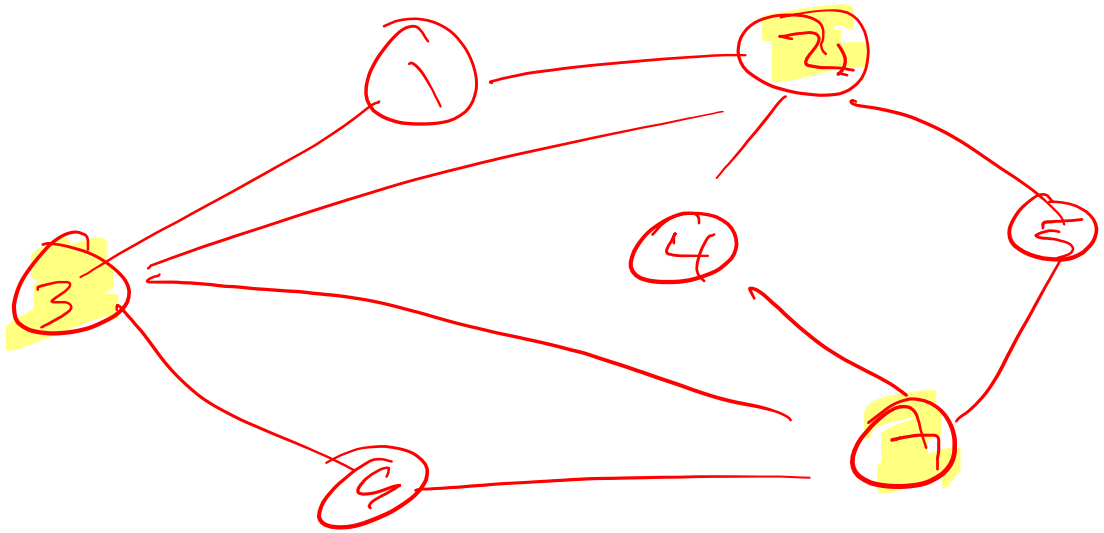
$\geq_P$ $\geq_P$

Uses for reduction X $\leq_P$ Y

- ***Constructive***: build an algorithm for X using Y
- ***Equivalence***: if can also show that Y $\leq_P$ X then problems are equivalent
- ***Intractability***: if we know Y is not solvable (efficiently), then X isn't either
- ***Evidence***: If lots of people tried to solve Y and failed, then you're unlikely to solve X

Independent set and Vertex cover

- **Def.** For a graph G = (V,E) a *vertex cover* is a set T ⊆ V for which every edge (u,v) ∈ E has u ∈ T or v ∈ T (or both)



Independent set and Vertex cover

- **Def.** For a graph G = (V,E) a *vertex cover* is a set T ⊆ V for which every edge (u,v) ∈ E has u ∈ T or v ∈ T (or both)

- Search problem: Given a graph G = (V,E), find a vertex cover of minimal size
- **Decision problem**: Given a graph G = (V,E) and number k, does G have a vertex cover of size k?

Vertex-Cover-Search $\leq_P$ Vertex-Cover-Dec

Independent set and Vertex cover

← Red means vertex cover solution

Def. For a graph G = (V,E) an *independent set* is a set S ⊆ V for which no u,v ∈ S have an edge

Def. For a graph G = (V,E) a *vertex cover* is a set T ⊆ V for which every edge (u,v) ∈ E has u ∈ T or v ∈ T (or both)

G T

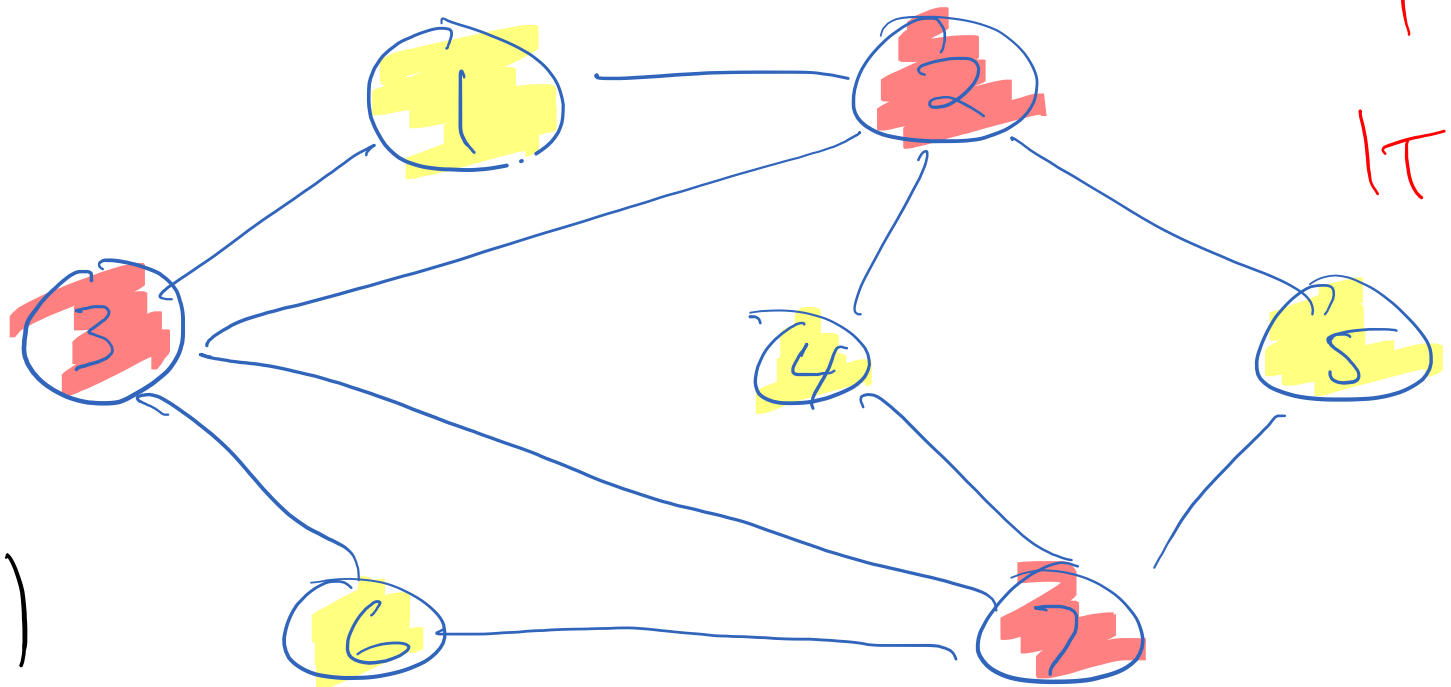
S = V − T

u, v ∈ S

s.t. e(u,v)

T = V − S

|T| = |V| − |S|



Independent set and Vertex cover

Lemma. Let G = (V,E) be a graph. Then S is an independent set if and only if its complement V - S is a vertex cover

Vertex-Cover-Dec $\leq_P$ Ind-Set-Dec

Ind-Set-Dec $\leq_P$ Vertex-Cover-Dec

Problems are polytime equivalent!

Does this mean we know how to solve either, efficiently?

Another example: set cover

- Decision problem: Given a set U of n elements, a collection $\boldsymbol{S} = \{S_1, S_2, \ldots, S_m\}$ of subsets of U, and a number k, does there exist a collection of at most k of these sets whose union is U?

Vertex-Cover-Dec $\leq_P$ Set-Cover-Dec

Vertex-Cover-Dec $\leq_P$ Set-Cov-Dec

Want:

Given a graph G = (V,E) and number k, does G have a vertex cover of size k?

Given:

Given a set U of n elements, a collection ***S*** = $\{S_1,S_2,\ldots,S_m\}$ of subsets of U, and a number k, does there exist a collection of at most k of these sets whose union is U?

Vertex-Cover-Dec $\leq_P$ Set-Cov-Dec

- Universe U = E
- For each v ∈ V, add subset S_v = { e ∈ E : e incident to v}

- **Lemma.** G has vertex cover of size k iff (U,S,k) contains set cover of size k

- What do we know about **Ind-Set-Dec** versus **Set-Cov-Dec**?

- Are the all these problems polytime equivalent?

One more example: SAT

- n boolean variables $X = x_1, \ldots, x_n$. Each is in {0,1}
- Clause C is a logical OR of subset of variables or their complements
  - E.g.: $(x_1 \vee \overline{x}_2)$

$\vee$	0	1
0	0	1
1	1	1

$\wedge$	0	1
0	0	0
1	0	1

- Boolean formula over these variables is logical AND of clauses
  - E.g., $(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_3) \wedge (x_2 \vee \overline{x}_3)$
- Truth assignment is a function $v : X \to \{0,1\}$
  - Satisfying if all clauses evaluate to 1

$x_1 = 0$
$x_2 = 1$
$x_3 = 0$

- **Def**. Given set of clauses $C_1, \ldots, C_k$ over $x_1, \ldots, x_n$ does there exist a satisfying truth assignment?

3SAT

- If all clauses of size 2 (called 2SAT), then linear time solution
- **3SAT**. Given set of clauses $C_1, \ldots, C_k$ each of length 3 over $x_1,\ldots,x_n$ does there exist a satisfying truth assignment?

$$(x_1 \vee \bar{x}_3 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$

$x_3 = 0$

3SAT-Dec $\leq_P$ Ind-Set-Dec

$x_1 = 1$

$x_3 = 1$

$x_4 = 1$

Want:

$C_1,\ldots,C_k$

$x_1,\ldots,x_n$

→ 3SAT decider (**3SAT-Dec**) → 0/1

Does formula have a satisfying assignment?

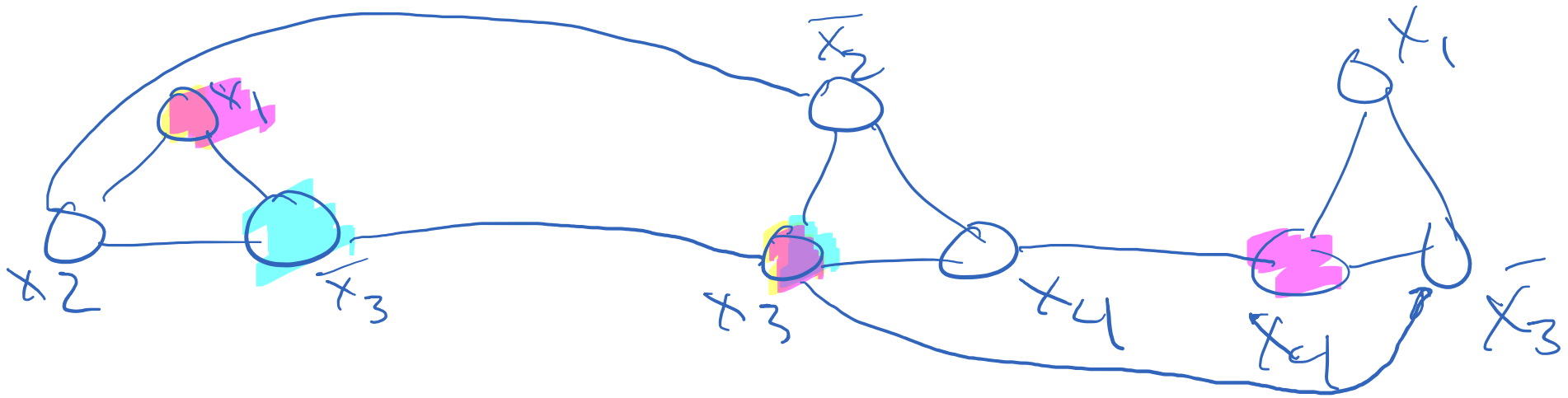
Given:

G,k

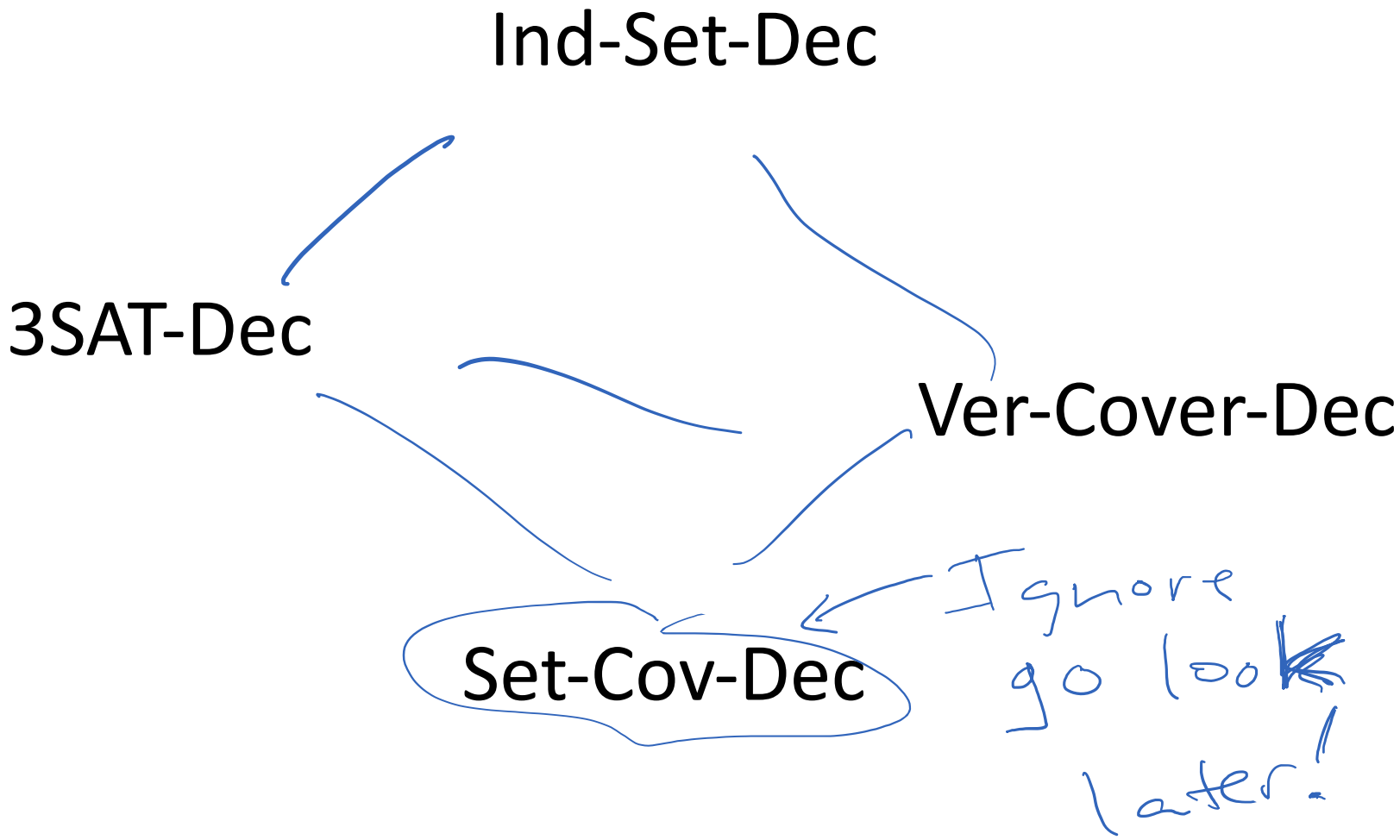
→ Independent set decision solver (**Ind-Set-Dec**) → 0/1

Does graph have an independent set of size k?

$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_4 \vee \bar{x}_3)$



Lots of problems and relations



The class P

- We let P be the set of all decision problem problems for which there exists a polytime algorithm solver

- We have seen lots of examples of problems in P

Is Ind-Set-Dec in P?

The class NP

$NP \subseteq P$ $NP \supseteq P$

- We let NP be the set of all decision problem problems for which there exists a polytime certifier
- A certifier for decision problem X is an algorithm B that:
  - B runs in polytime in inputs s, t
  - There is a polynomial function p so that for every string s, we have $s \in X$ iff there exists string t such that $|t|$ is polynomial function of $|s|$ and $B(s,t) = 1$

- In words: we can ***check*** a solution in polytime, but not necessarily ***find*** it in polytime
- **Example:** Ind-Set-Dec certificate would be list of k vertices in supposed independent set. Just do $O(k^2)$ check of each pair

An (at least) million dollar open problem

Prove that $P \neq NP$ -or- Prove that $P = NP$

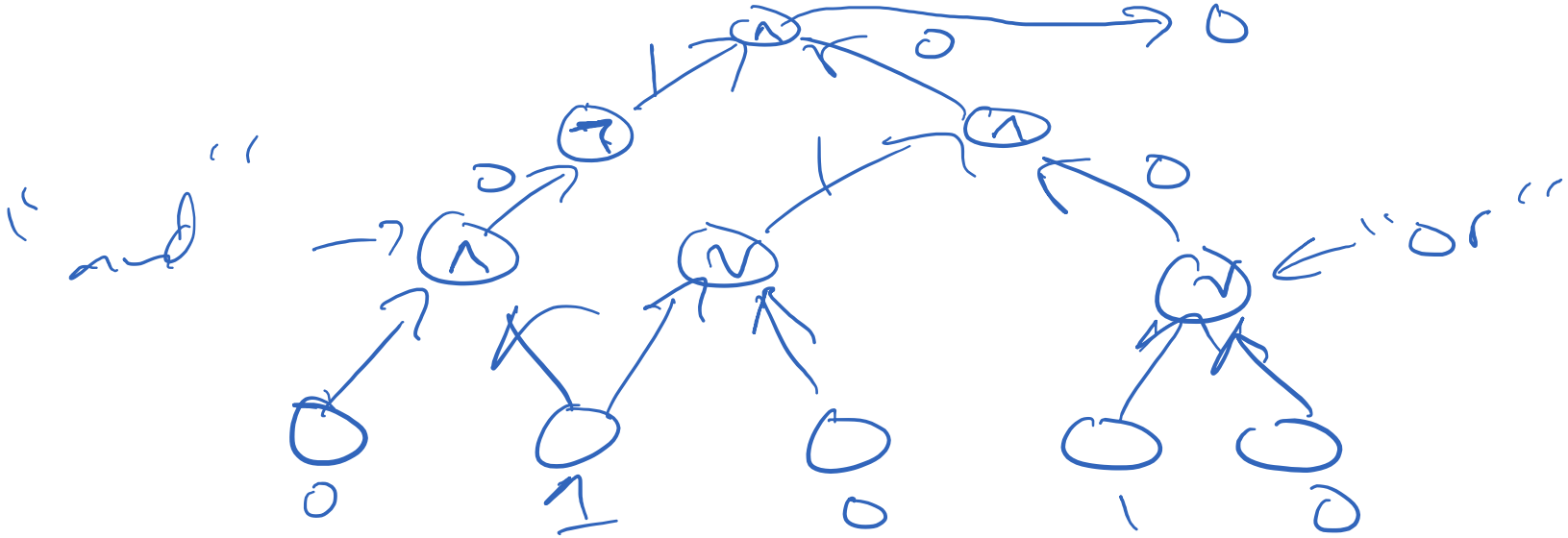
NP-completeness

- Can we find problems that are as hard as ***all problems*** in NP?
- **Def**. A problem X is NP-complete if
  - $X \in NP$
  - For all $Y \in NP$, $Y \leq_P X$
- Is it clear whether an X exists that is NP-complete?
- Say we can show some X is NP-complete. And we show a polytime algorithm for X. Then what?

Proven that $P = NP$

An NP-complete problem: circuit satisfiability

- (Boolean) circuit K is a labeled, directed, acyclic graph with:
  - Sources in K are labeled either with constants 0 or 1. Called inputs
  - Other nodes labeled with a Boolean operation in {∧,∨, ¬}
    - Examples: $0 \wedge 1 = 0$ $0 \vee 1 = 1$ $\neg 1 = 0$
  - Single node with no outgoing edges is the output



An NP-complete problem: circuit satisfiability

- Circuit satisfiability (CircSat) is NP-complete proof sketch
  - Easy to show that CircSat ∈ NP
- Consider any decision problem Y ∈ NP. Show that Y $\leq_P$ CircSat
  - Since Y is in NP, we have polytime algorithm to check certificates, i.e., a circuit K with input Y-input s and certificate t
  - To solve arbitrary Y instance s, let K' be K but with input s hard-coded and leaves as variables the certificate t
  - Run CircSat solver on K'. Output its answer as solution for Y instance

An NP-complete problem: circuit satisfiability

- Circuit satisfiability (CircSat) is NP-complete proof sketch
  - Easy to show that CircSat ∈ NP
- Consider any decision problem Y ∈ NP. Show that Y $\leq_P$ CircSat
  - Since Y is in NP, we have polytime algorithm to check certificates, i.e., a circuit K with input Y-input s and certificate t
  - To solve arbitrary Y instance s, let K' be K but with input s hard-coded and leaves as variables the certificate t
  - Run CircSat solver on K'. Output its answer as solution for Y instance

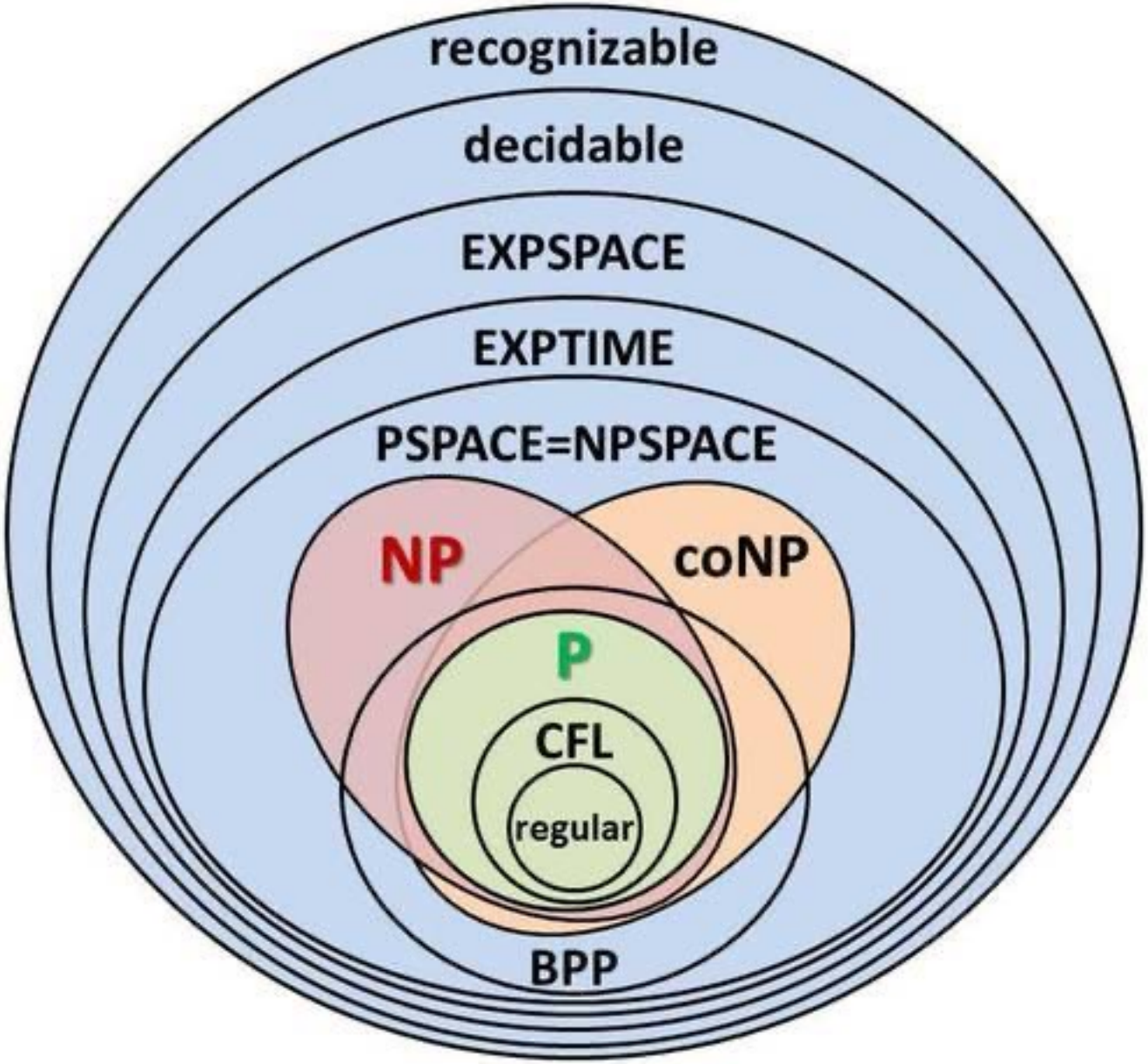
Showing that Ind-Set-Dec is NP-complete

- Recall: transitivity of polytime reducibility

$$Y \leq_P \text{CircSat} \leq_P \text{3SAT-Dec} \leq_P \text{Ind-Set-Dec}$$

- **Corollary**: If Y is an NP-complete problem, and X is a problem in NP with $Y \leq_P X$, then X is NP-complete

- We can translate a circuit to a 3SAT formula efficiently. See textbook


recognizable
decidable
EXPSPACE
EXPTIME
PSPACE=NPSPACE
NP
coNP
P
CFL
regular
BPP

Discussion

- Perhaps simple seeming goal: fast algorithm for finding independent sets
  - Brute-force algorithm is exponential
  - Can we do better? When do we give up solving it?
- NP-completeness gives us powerful tool to take advantage of other peoples' hard work trying to solve (other) problems
  - It's why I knew you all didn't solve the problem efficiently with your (very good!) suggestions
- But practically it's rarely an option to ***not solve*** a problem
  - Next week we'll see what people do. Change or relax the problem