# CS 580
# ALGORITHM DESIGN AND ANALYSIS

## NP and NP-completeness

Vassilis Zikas

# SO FAR

- Defined Cook reductions
- Saw some simple reduction strategies
- 3-SAT$\leq_P$ INDEPENDENT SET $\leq_P$VERTEX COVER $\leq_P$ SET COVER

- Today:
  - The class NP! (8.3 in KT)
  - NP-completeness
  - co-NP

# MORE REDUCTION STRATEGIES

- Decision problem:
  - Does there exist a vertex cover of size $\leq k$?
- Search problem:
  - Find the vertex cover of minimum cardinality
- Self-reducibility:
  - Search version $\leq_P$ decision version
  - Applies to all NP-complete problems
  - Justifies our focus on decision problems

# MORE REDUCTION STRATEGIES

- Self-reducibility for vertex cover
  - Binary search for cardinality $k^*$ of min vertex cover
  - Find a vertex $v$ such that $G - \{v\}$ has a vertex cover of size $k^* - 1$
    - Any vertex in any min vertex cover will have this property
  - Include $v$ in the vertex cover
  - Recursively find a min vertex cover in $G - \{v\}$

# 8.3: DEFINITION OF NP

# DECISION PROBLEMS

- Decision problem
  - $X$ is a set of strings
  - Instance: string $s$
  - Algorithm $A$ solves problem $X$:
    - $A(s) = yes$ if and only if $s \in X$

- Algorithm $A$ runs in polynomial time if for every string $s$, $A(s)$ terminates in at most $p(|s|)$ steps, where $p(.)$ some polynomial

- PRIMES: $\{2, 3, 5, 7, 11, 13, 17, 23, 29, 31, \ldots\}$
  - Algorithm: [Agrawal et al. 2002] $p(|s|) = s^8$

# DEFINITION OF P

- **P**: Decision problems for which there is a poly-time algorithm.

| Problem | Description | Algorithm | Yes | No |
|---|---|---|---|---|
| MULTIPLE | Is x a multiple of y? | Grade school division | `51, 17` | `51, 16` |
| RELPRIME | Are x and y relatively prime? | Euclid (300 BCE) | `34, 39` | `34, 51` |
| PRIMES | Is x prime? | AKS (2002) | `53` | `51` |
| EDIT-DISTANCE | Is the edit distance between x and y less than 5? | Dynamic programming | `niether` `neither` | `acgggt` `ttttta` |
| LSOLVE | Is there a vector x that satisfies Ax = b? | Gauss-Edmonds elimination | $\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ |

# CERTIFICATES

- *Finding* a solution versus *checking* a solution
- We do not know good algorithms that decide whether a CNF formula is satisfiable
- But, given a proposed solution we can quickly check if it satisfies the formula or not
- Given a CNF formula one can provide evidence that the formula is satisfiable
  - Evidence = satisfying assignment
- What evidence is there that a formula is **not** satisfiable?
- Formalizing "evidence" is going to be crucial for us

# CERTIFIER

- An algorithm $C(s, t)$ is an efficient certifier for a problem $X$ if
  - $C$ runs in polynomial time
  - For every string $s$, we have $s \in X$ if and only if there exists a string $t$ such that $C(s, t) = yes$
- $t$ is called the "certificate" or "witness"

# NP

**NP** (nondeterministic polynomial time) is the set of all problems for which there exists an efficient certifier

# CERTIFIERS AND CERTIFICATES: 3-SAT

- SAT:
  - Given a CNF formula $\phi$ is there a satisfying assignment?
- Certificate:
  - An assignment of truth values to the $n$ Boolean variables
- Certifier:
  - Check that each clause in $\phi$ has at least one true literal

- Example:

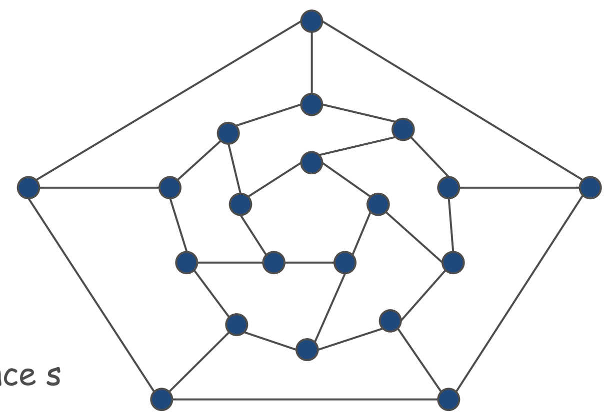
$$\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_1 \vee x_2 \vee x_4\right) \wedge \left(\overline{x_1} \vee \overline{x_3} \vee \overline{x_4}\right)$$

Certificate $t$: $\quad x_1 = 1,\; x_2 = 1,\; x_3 = 0,\; x_4 = 1$

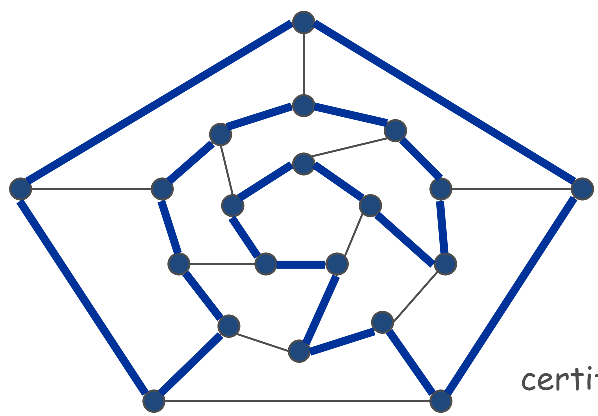
- Conclusion: SAT is in NP

# CERTIFIERS AND CERTIFICATES: HAMILTON CYCLE

- HAM-CYCLE: Given an undirected graph $G$, does there exist a simple cycle $C$ that visits every node?
- Certificate: Ordered list of nodes
- Certifier: Check that the ordered list contains each node in $V$ exactly once, and that there is an edge between adjacent nodes in the list (and between the first and last node).
- Conclusion: HAM-CYCLE in NP



instance s



certificate t

# P, NP, EXP

- P: Decision problems for which there is poly-time algorithm
- NP: Decision problems for which there is a poly-time certifier
- EXP: Decision problems for which there is an exponential time algorithm

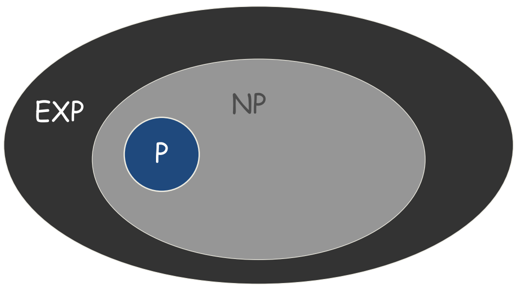
- Claim: $P \subseteq NP$
  - Proof: Consider any problem $X$ in P
  - By definition, there exists a poly-time algorithm $A(s)$ that solves $X$
  - Certificate $t = \emptyset$, certifier $C(s, t) = A(s)$

- Claim: $NP \subseteq EXP$
  - Proof: Consider any problem $X$ in NP
  - By definition, there exists a poly-time certifier $C(s, t)$ for $X$
  - To solve input $s$, run $C(s, t)$ on all strings $t$ with $t \leq p(|s|)$
  - Return *yes* if $C(s, t)$ returns *yes* for any of these

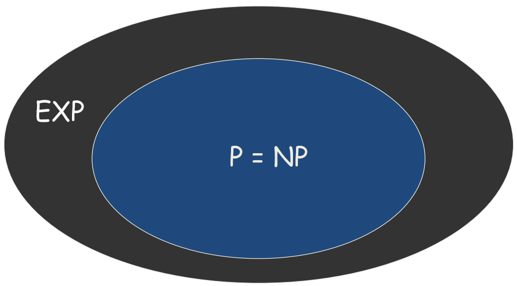
# THE QUESTION

- Does P=NP? [Cook 1971, Edmonds, Levin, Yablonksi, Godel]
- Is a solving a problem easier than verifying a solution?
    - Easiest way to become the most famous computer scientist alive!
    - Clay prize: 1 million dollars

# P VERSUS NP




If $P \neq NP$




If $P = NP$

# P VERSUS NP

- If $P = NP$ the economy will probably collapse
  - E.g. RSA is no longer secure, and therefore our bank accounts are no longer secure
- If $P \neq NP$, no efficient algorithms for 3-SAT, INDEPENDENT SET, SET COVER,…
- Consensus: most people believe that $P \neq NP$

# NP-COMPLETENESS (8.4 IN KT)

- Cook reduction: Problem $X$ polynomial reduces to problem $Y$ if arbitrary instances of $X$ can be solved using:
  - Polynomial number of standard steps, and
  - Polynomial number of queries to an oracle for $Y$
- Karp reduction: Problem X polynomial reduces to problem $Y$ if given any input $x$ of $X$ we can construct an input $y$ of $Y$, such that $x$ is a *yes* instance of $X$ if and only if $y$ is a *yes* instance of $Y$

# NP-COMPLETENESS

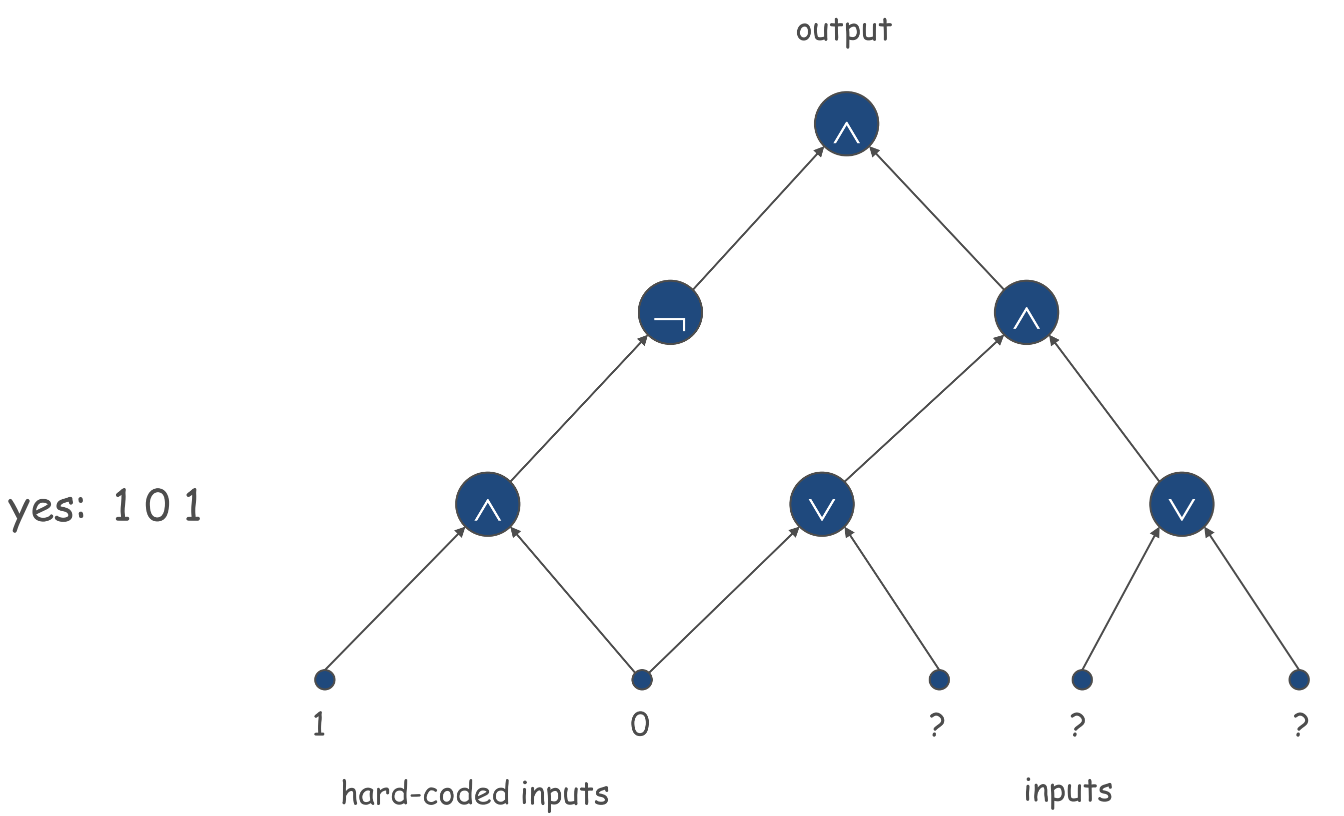
- Most reductions we've seen so far are actually Karp reductions
  - Only one call at the oracle for $Y$, at the end
- Cook and Karp reductions are not known to be the same with respect to NP
  - In a Cook reduction, we call the oracle multiple times, perhaps adaptively
- If they are the same, then $co$-NP $=$ NP
- If they are not the same then $P \neq NP$
- We will mostly focus on Karp reductions, since they are the standard type

# NP-COMPLETENESS

- Definition: A problem $Y$ is NP-hard, if for every $X$ in NP, $X \leq_P Y$
- Definition: A problem $Y$ in NP is NP-complete, if for every $X$ in NP, $X \leq_P Y$
- Theorem: Suppose $Y$ is an NP-complete problem. Then $Y$ is solvable in polynomial time if and only if P=NP
- Proof:
  - $\Longleftarrow$ If P=NP $Y$ can be solved in polytime since it's in $NP$
  - $\Longrightarrow$ Suppose $Y$ is solvable in polynomial time. Let $X$ be an arbitrary problem in $NP$. Since $X \leq_P Y$ we can solve $X$ in polytime, thus $NP \subseteq P$. We already know that $P \subseteq NP$
- Fundamental question: Are there "natural" NP-complete problems?

# CIRCUIT SATISFIABILITY

- CIRCUIT-SAT: Given a combinational circuit built out of AND, OR and NOT gates, is there a way to set the circuit inputs so that the output is 1?



# THE "FIRST" NP-COMPLETE PROBLEM

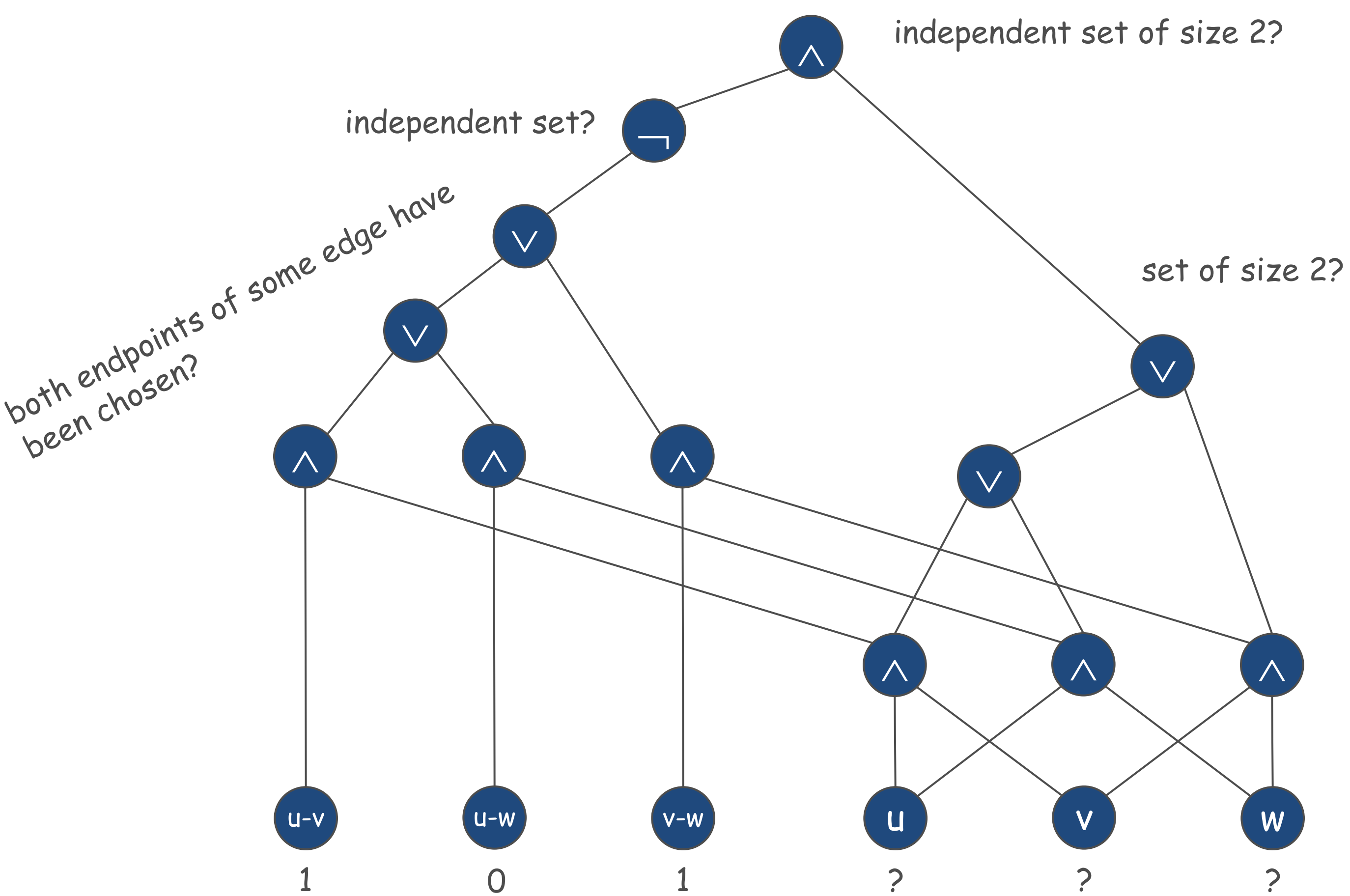
- Theorem: CIRCUIT-SAT is NP-complete!
  - [Cook 1971, Levin 1973]
- Proof (sketch):
  - Any algorithm that takes a fixed number of bits $n$ as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.
  - Natural: Algorithms are implemented on computers (i.e. on literal circuits)
    - Sketchy part of proof: Algorithms typically have no trouble dealing with inputs of larger size (i.e. more than $n$ bits), as opposed to circuits.

# THE "FIRST" NP-COMPLETE PROBLEM

- Consider some problem $X$ in NP, with a polytime certifier $C(s, t)$
- To determine whether $s$ is in $X$ need to know if there exists a $t$ of length $p(|s|)$ such that $C(s, t) = true$
- View $C(s, t)$ as an algorithm on $|s| + p(|s|)$ bits, and convert it into a circuit $K_s$
  - First $|s|$ bits are hard-coded to $s$
- $K_s$ satisfiable if and only if there exists a length $p(|s|)$ input string such that $C(s, t) = true$

# EXAMPLE

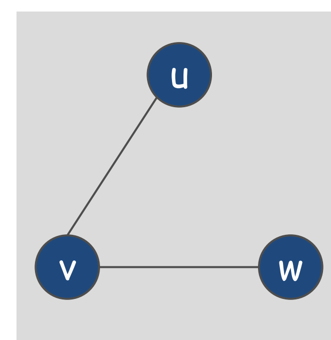
- $K_s$ satisfiable iff there exists an IS of size 2




$\binom{n}{2}$ hard-coded inputs (graph description)

n inputs (nodes in independent set)

G = (V, E), n = 3



# ESTABLISHING NP-COMPLETENES

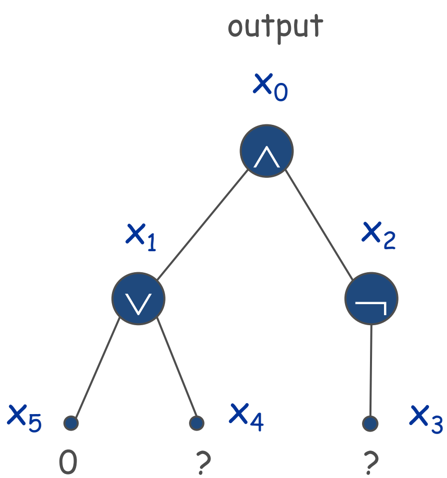
- Once we have the first NP-Complete problem, others fall like dominoes
  - We have hundreds (if not thousands) of known NP-complete problems!
- Recipe to establish NP-completeness for $X$
  - Step 1: Show that $X$ is in NP (the step that most students forget)
  - Step 2: Choose an NP-complete problem $Y$
  - Step 3: Prove that $Y \leq_P X$
- Proof: Let $W$ be any problem in $NP$
  - $W \leq_P Y \leq_P X$
  - Hence $X$ is NP-complete

# THEOREM: 3-SAT IS NP-COMPLETE

- Proof:
  - We've seen that 3-SAT is in NP, so it suffices to show that CIRCUIT-SAT $\leq_P$ 3-SAT
  - Let $K$ be any circuit
  - Create a 3-SAT variable $x_i$ for every element of $K$
    - $x_2 = \neg x_3 \rightarrow$ add two clauses $(x_2 \vee x_3)$, $(\overline{x_2} \vee \overline{x_3})$
      - Can't set both to true or both to false!
    - $x_1 = x_4 \vee x_5 \rightarrow$ add three clauses $(x_1 \vee \overline{x_4})$, $(x_1 \vee \overline{x_5})$, $(\overline{x_1} \vee x_4 \vee x_5)$
      - If $x_1$ is true, at least one of $x_4$ and $x_5$ has to be true. If $x_1$ if false, both $x_4$ and $x_5$ have to be false
    - $x_1 = x_4 \wedge x_5 \rightarrow$ add three clauses $(x_4 \vee \overline{x_1})$, $(x_5 \vee \overline{x_1})$, $(x_1 \vee \overline{x_4} \vee \overline{x_5})$
      - If $x_1$ is true, at both of $x_4$ and $x_5$ have to be true. If $x_1$ if false, at least one of $x_4$ and $x_5$ has to be false

# THEOREM: 3-SAT IS NP-COMPLETE

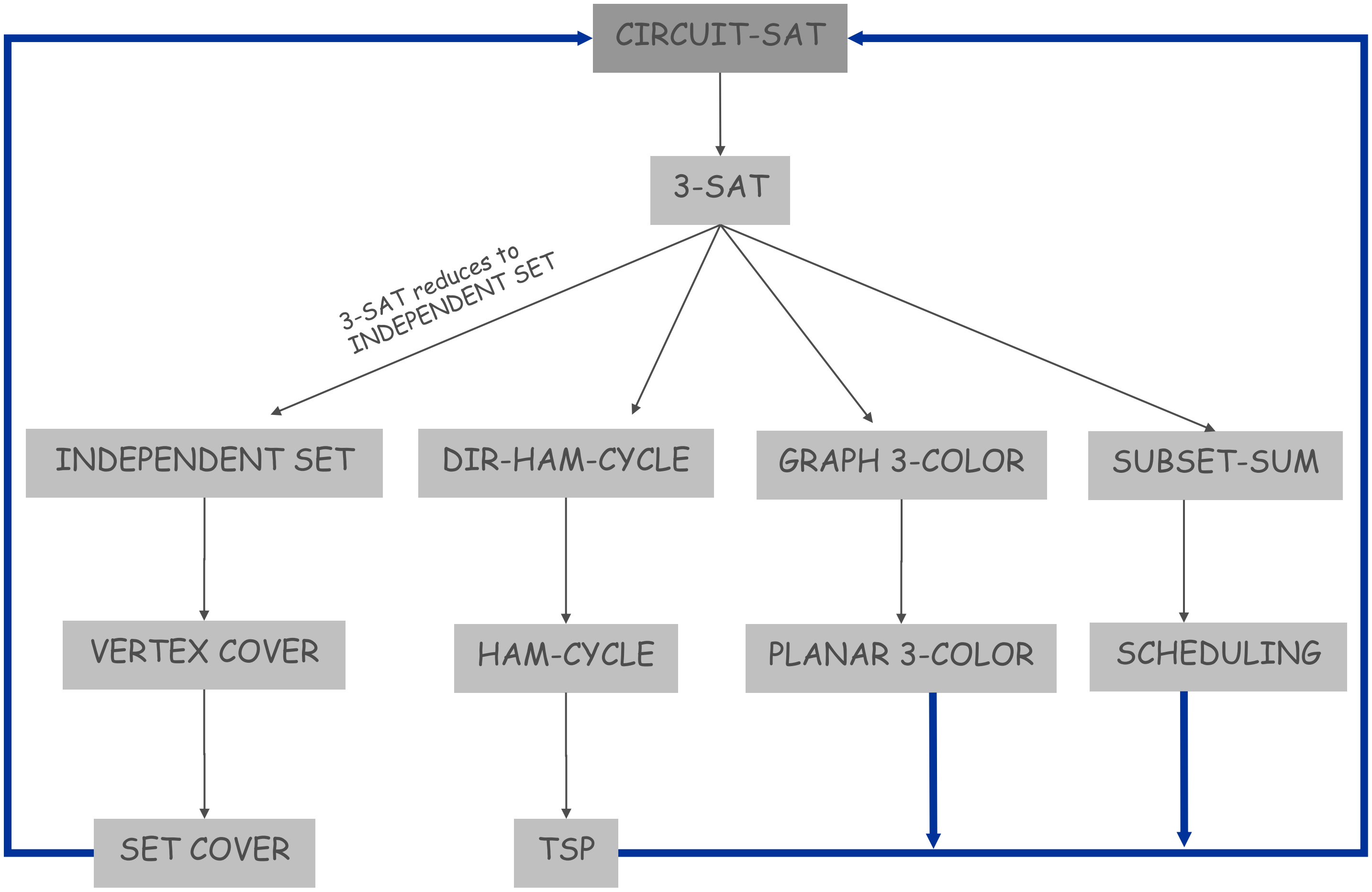
- Proof (continued):
  - Hard-coded input and output values
    - $x_5 = 0 \rightarrow$ add clause $(\overline{x_5})$
    - $x_1 = 1 \rightarrow$ add clause $(x_1)$




$(\overline{x_5}) \wedge (x_1 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_5}) \wedge (\overline{x_1} \vee x_4 \vee x_5) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_0}) \wedge (x_2 \vee \overline{x_0}) \wedge (x_0 \vee \overline{x_1} \vee \overline{x_2})$

# NP-COMPLETENESS

- Observation. All problems below are NP-complete and polynomial reduce to one another!



# NP-COMPLETE PROBLEMS

- Basic big “genres” of NP-complete problems:
  - Packing: SET-PACKING, INDEPENDENT SET
  - Covering: SET-COVER, VERTEX-COVER
  - Constraint Satisfaction: 3-SAT, CIRCUIT-SAT
  - Partitioning: COLORING, 3-D MATCHING
  - Numerical problems: SUBSET-SUM, KNAPSACK
  - Sequencing: HAMILTONIAN CYCLE, TSP

# NP-COMPLETE PROBLEMS

- In practice, most problems are either in P or known to be NP-complete
- There are important exceptions:
  - Factoring
  - Graph isomorphism
  - Finding a Nash-equilibrium

# EXTENT AND IMPACT

- “Prime intellectual export of CS to other disciplines”
- Thousands of citations every year
  - More than “compilers” or “database”, etc
- Broad applicability and classification power
- “Captures vast domains of computational, scientific, mathematical endeavors, and seems to roughly delimit what mathematicians and scientists had been aspiring to compute feasibly.”
- A guide to scientific inquiry
  - E.g. Ising’s phase transition model for 3 dimensions
  - Top theoreticians searched for an analytical three-dimensional solution for many decades
  - Istrail showed that the problem is NP-complete in 2000.

# CO-NP AND THE ASYMMETRY OF NP

- Important detail about NP: We only need certificates for *yes* instances

- Example 1: SAT versus TAUTOLOGY
  - Can prove that a CNF formula is satisfiable by giving a satisfying assignment
  - How could you prove that a formula is not satisfiable?

- Example 2: HAM-CYCLE versus NO-HAM-CYCLE
  - Can prove that a graph is Hamiltonian by giving a Hamiltonian cycle
  - How could we prove that a graph is not Hamiltonian?

# CO-NP AND THE ASYMMETRY OF NP

- NP: Decision problems for which there is a poly-time certifier
- Definition: Given a decision problem $X$ its complement $\bar{X}$ is the same problem, but with $yes$ and $no$ instances reversed
  - I.e. $s \in \bar{X}$ iff $s \notin X$
- co-NP: Decision problems whose complement is in NP

# NP VERSUS CO-NP

- Fundamental question: Does NP = co-NP?
  - Do *yes* instances have succinct certificates iff *no* instances have succinct certificates?
  - Probably not…
- Theorem: If $NP \neq co\text{-}NP$ then $P \neq NP$
  - Proof idea:
  - $P$ is closed under taking complements
  - If $P = NP$, then $NP$ is closed under taking complements
  - Therefore, $NP = co\text{-}NP$
  - This is the contrapositive of the theorem

# PROBLEMS NP ∩ CO-NP

- Example: Given a bipartite graph, does it have a perfect matching:
  - If $yes$: can exhibit a perfect matching
  - If $no$: can exhibit a set of nodes $S$, s.t. $|N(S)| < |S|$
- Every problem in $NP \cap co\text{-}NP$ has a "good characterization"
  - Short proof for $yes$ and short proof for $no$
- Observation: $P \subseteq NP \cap co\text{-}NP$
- Fundamental open question: Does P equal NP ∩ co-NP?
  - Mixed opinions
  - Many examples where problems had good characterizations first, and then years later an efficient algorithm was developed
    - E.g. linear programming, primality testing
- E.g. Factoring is in $NP \cap co\text{-}NP$ but not known to be in P

# SUMMARY

- Definition of NP (8.3 in KT)
- NP-Completeness (8.4 in KT)
- co-NP (8.9 in KT)