Intro to Theory of Computation

CS 464

LECTURE 26

Last time

- Polynomial-time reductions

Today

- Polynomial-time reductions
- NP-completeness

Adam Smith, Sofya Raskhodnikova

Classify Problems

- **Desiderata:** classify problems according to those that can be solved in polynomial-time and those that cannot.
- Some problems *provably require exponential time* (next month):
  - Given a Turing machine, does it halt in at most *k* steps?
  - Given a board position in an *n*-by-*n* generalization of chess, can black guarantee a win?
- **Frustrating news:** huge number of fundamental problems have defied classification for decades.
- **Chapters 7.4-7.5 (NP-completeness):** Show that these fundamental problems are "computationally equivalent" and appear to be different manifestations of one **really hard** problem.

Polynomial-time reduction

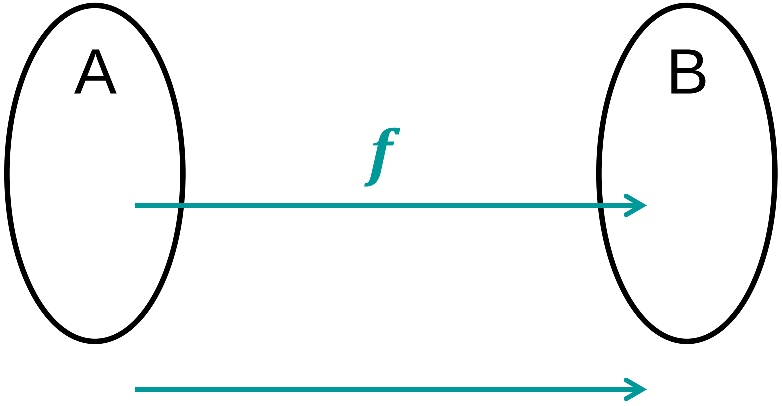
Given languages A and B,

$A \leq_p B$

if there is a ***poly-time*** computable function f,

such that for all strings w,

$w \in A$ iff $f(w) \in B$.



Implication of poly-time reductions

Theorem. If A $\leq_p$ B and B$\in$ **P** then A$\in$ **P**.

(So, if A $\leq_p$ B and A$\notin$ **P** then B$\notin$ **P**.)

Theorem. If A $\leq_p$ B and B $\leq_p$ C then A $\leq_p$ C.

(Poly-time reductions compose.)

Reduction by simple equivalence

Basic reduction strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

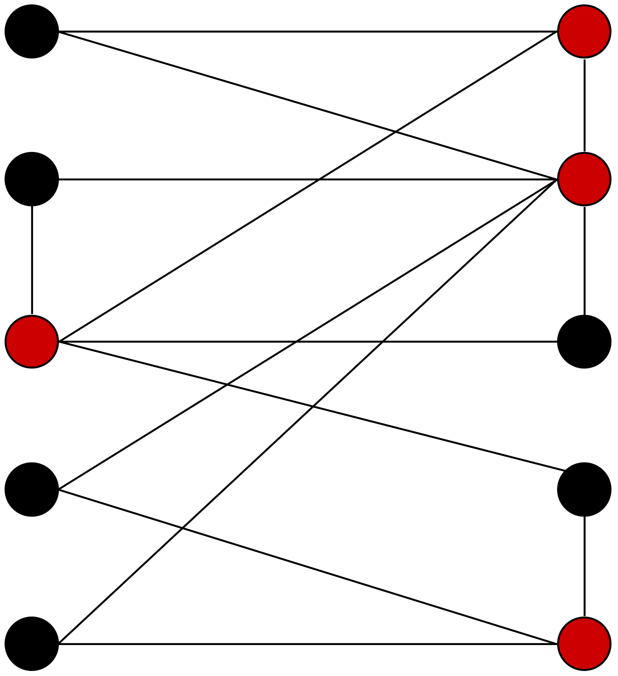
Independent Set

Given an undirected graph G, an **independent set** in G is a set of nodes, which includes at most one endpoint of every edge.

INDEPENDENT SET $= \{\langle G, k\rangle \mid G$ is an undirected graph which has an independent set with k nodes$\}$

- Is there an independent set of size ≥ 6?
  - Yes.
- Is there an independent set of size ≥ 7?
  - No.

● independent set



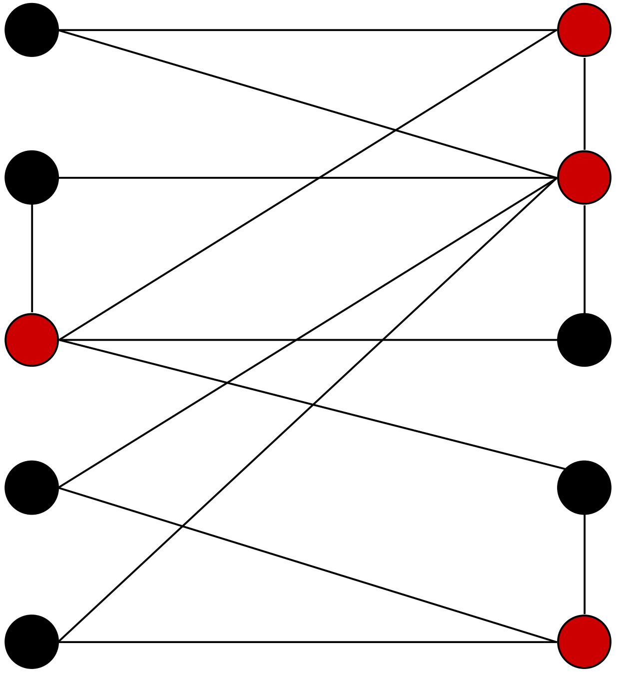
Vertex Cover

Given an undirected graph G, a **vertex cover** in G is a set of nodes, which includes at ***least*** one endpoint of every edge.

VERTEX COVER$=\{\langle G,k\rangle \mid G$ is an undirected graph which has a vertex cover with k nodes$\}$

- Is there vertex cover of size ≤ 4?
  - Yes.

vertex cover



- Is there a vertex cover of size ≤ 3?
  - No.

Set Cover

Given a set U, called a *universe*, and a collection of its subsets $S_1, S_2, \ldots, S_m$, a **set cover** of U is a subcollection of subsets whose union is U.

- SET COVER=$\{\langle U, S_1, S_2, \ldots, S_m; k\rangle \mid$ U has a set cover of size $k\}$

U = { 1, 2, 3, 4, 5, 6, 7 }

k = 2

S_1 = {3, 7}	S_4 = {2, 4}
S_2 = {3, 4, 5, 6}	S_5 = {5}
S_3 = {1}	S_6 = {1, 2, 6, 7}

- Sample application.
  - m available pieces of software.
  - Set U of n capabilities that we would like our system to have.
  - The ith piece of software provides the set $S_i \subseteq$ U of capabilities.
  - Goal: achieve all n capabilities using fewest pieces of software.

CS 464

Satisfiability

- **Boolean variables:** variables that can take on values T/F (or 1/0)
- **Boolean operations:** $\vee$, $\wedge$, and $\neg$
- **Boolean formula:** expression with Boolean variables and ops

SAT = {⟨Φ⟩ | Φ is a satisfiable Boolean formula}

- **Literal:** A Boolean variable or its negation. x_i or $\overline{x_i}$
- **Clause:** OR of literals. $C_j = x_1 \vee \overline{x_2} \vee x_3$
- **Conjunctive normal form (CNF):** AND of clauses. $\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

3SAT = {⟨Φ⟩ | Φ is a satisfiable Boolean CNF formula, where each clause contains exactly 3 literals}

each corresponds to a different variable

Ex: $(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$

Yes: x_1 = true, x_2 = true x_3 = false.

Summary of reduction strategies

- Basic reduction strategies.
  - Simple equivalence: INDEPENDENT-SET $\equiv_P$ VERTEX-COVER.
  - Special case to general case: VERTEX-COVER $\leq_P$ SET-COVER.
  - Encoding with gadgets: 3-SAT $\leq_P$ INDEPENDENT-SET.

- Composition. If $X \leq_P Y$ and $Y \leq_P Z$, then $X \leq_P Z$.
- Thus,

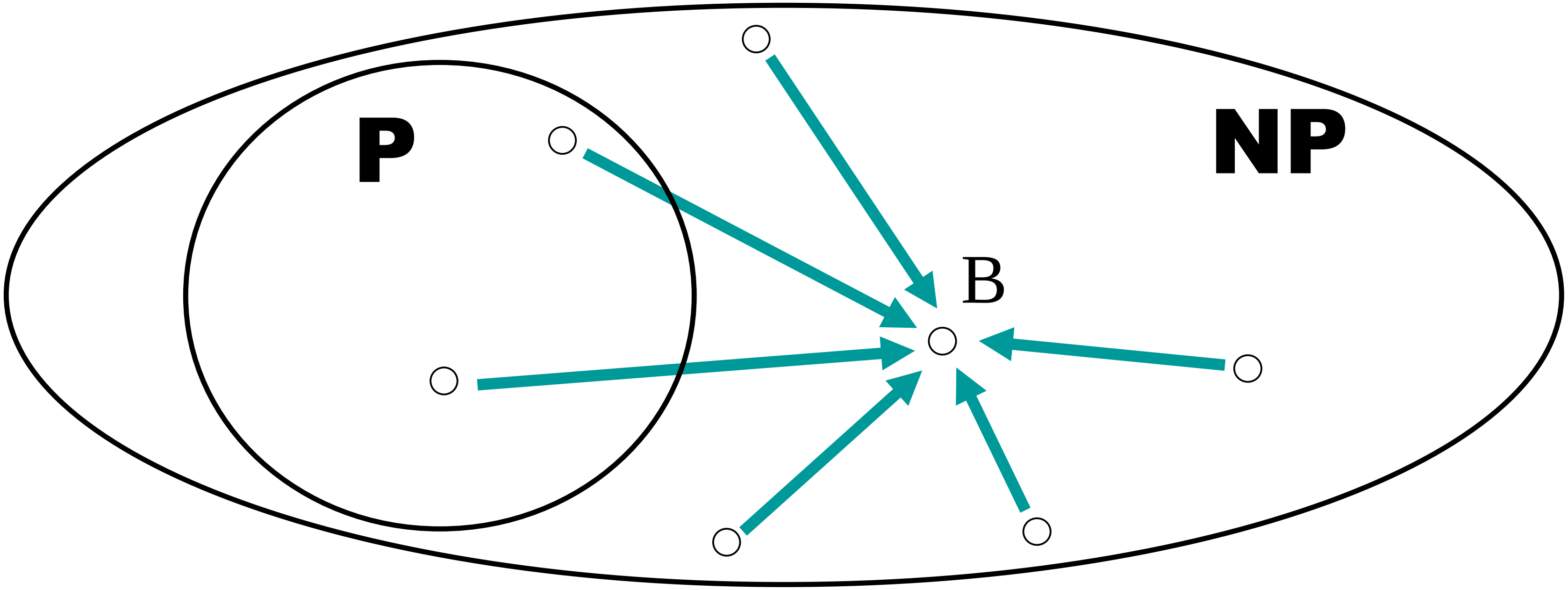
3SAT $\leq_P$ INDEPENDENT-SET $\leq_P$ VERTEX-COVER $\leq_P$ SET-COVER.

Are we done now?

Hardest problems in NP

A language B is **NP-complete** if

1. $B \in$ **NP**
2. B is **NP-hard**,i.e., every language in NP is poly-time reducible to B.

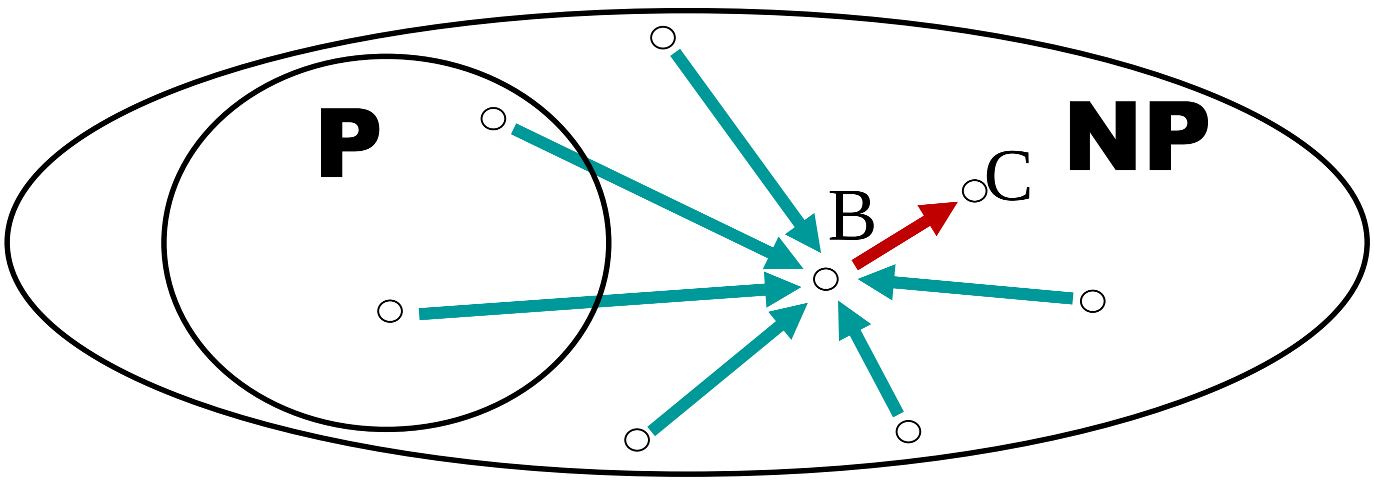


Implication of poly-time reductions

Theorem. If

- B is **NP**-complete,
- C∈ **NP** and
- $B \leq_p C$

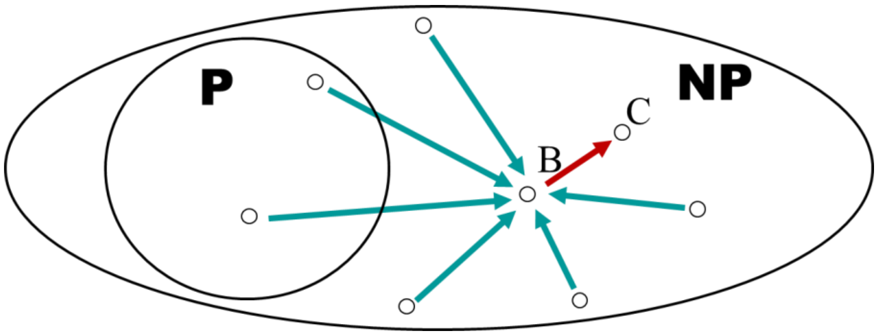
then C is **NP**-complete.



Implication of poly-time reductions

Theorem. If
- B is **NP**-complete,
- C$\in$ **NP** and
- B$\leq_p$C

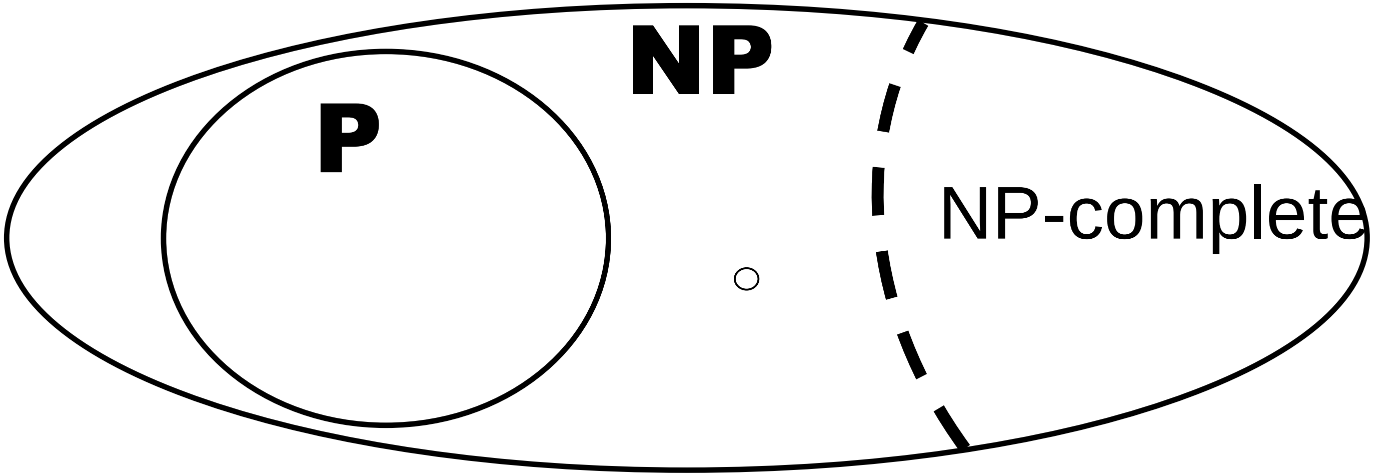
then C is **NP**-complete.



Theorem. If B is **NP**-complete and B$\in$ **P** then

$$\mathbf{P} = \mathbf{NP}.$$

(So, if B is **NP**-complete and $\mathbf{P} \neq \mathbf{NP}$ then there is no poly-time algorithm for B.)


NP
P
NP-complete

CS 464

An NP-complete problem

BA_{NTM} = {⟨M,x,$\underline{t}$⟩ | M is an NTM that accepts x in at most t steps}

Technical detail: $\underline{n}$ denotes 1^n.

Theorem. BA_{NTM} is NP-Complete.

1. $BA_{NTM} \in NP$:

   The list of guesses M makes to accept x in t steps is the certificate that $\langle M,x,\underline{t}\rangle \in BA_{NTM}$.

2. For all $A \in NP$, $A \leq_P BA_{NTM}$.

   $A \in NP$ iff there is an NTM N for A that runs in time $O(n^k)$.
   Let $f_A(w) = \langle N,w,\underline{c\,|w|^k}\rangle$.
   $\langle N,w,\underline{c\,|w|^k}\rangle \in BA_{NTM}$ iff N accepts w, iff $w \in A$.

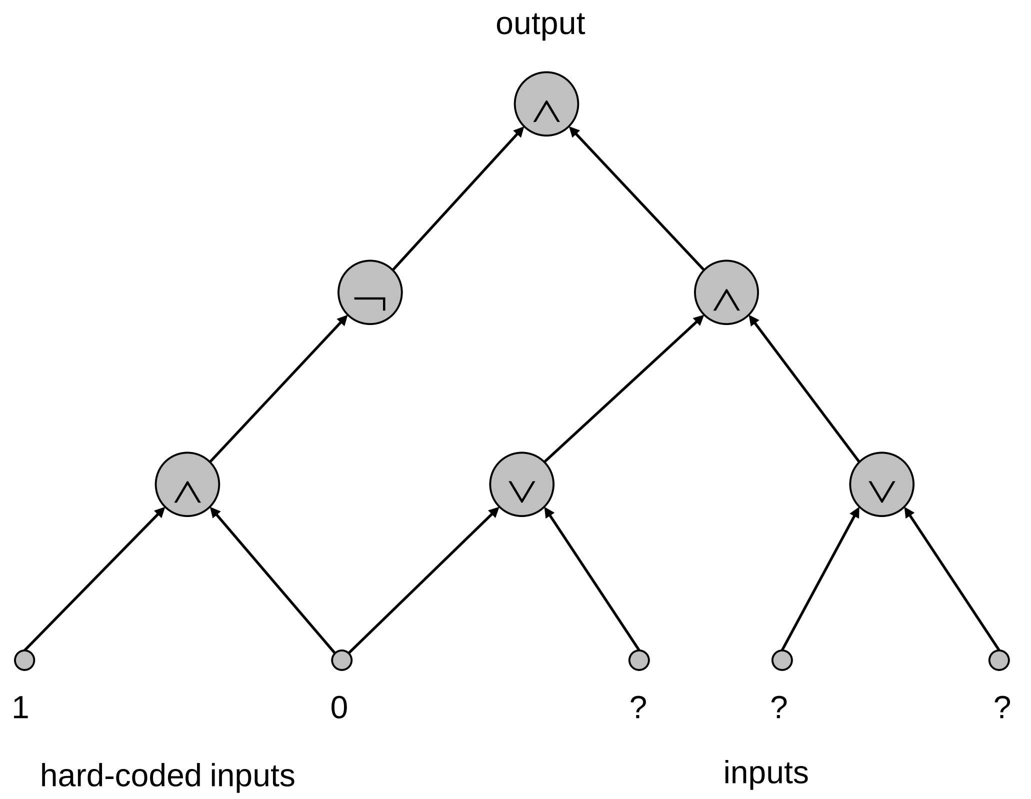
Circuit Satisfiability

A **circuit** is built out of AND, OR and NOT gates.

A circuit is **satisfiable** if one can set the circuit inputs, so that the output is 1.

CIRCUIT-SAT = $\{\langle C\rangle \mid C$ is a satisfiable circuit$\}$.

yes: 1 0 1



The "First" NP-Complete Problem

Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]

Proof sketch.

1. **CIRCUIT-SAT is in NP**

    certificate: input on which circuit is 1.

The "First" NP-Complete Problem

Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]

Proof sketch.

2. **For all A $\in$ NP, A $\leq_P$ CIRCUIT-SAT.**

- A TM that takes a fixed number of bits as input can be represented by a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.
- Consider some problem A $\in$ NP. It has a poly-time verifier V(w, c). To determine whether w$\in$A, need to know if there exists a certificate c of length p(|w|) such that V(w, c) accepts.
- View V(w, c) as an algorithm on |w| + p(|c|) bits (input w, certificate c) and convert it into a poly-size circuit C.
  - first |w| bits are hard-coded with w
  - remaining p(|s|) bits represent bits of t

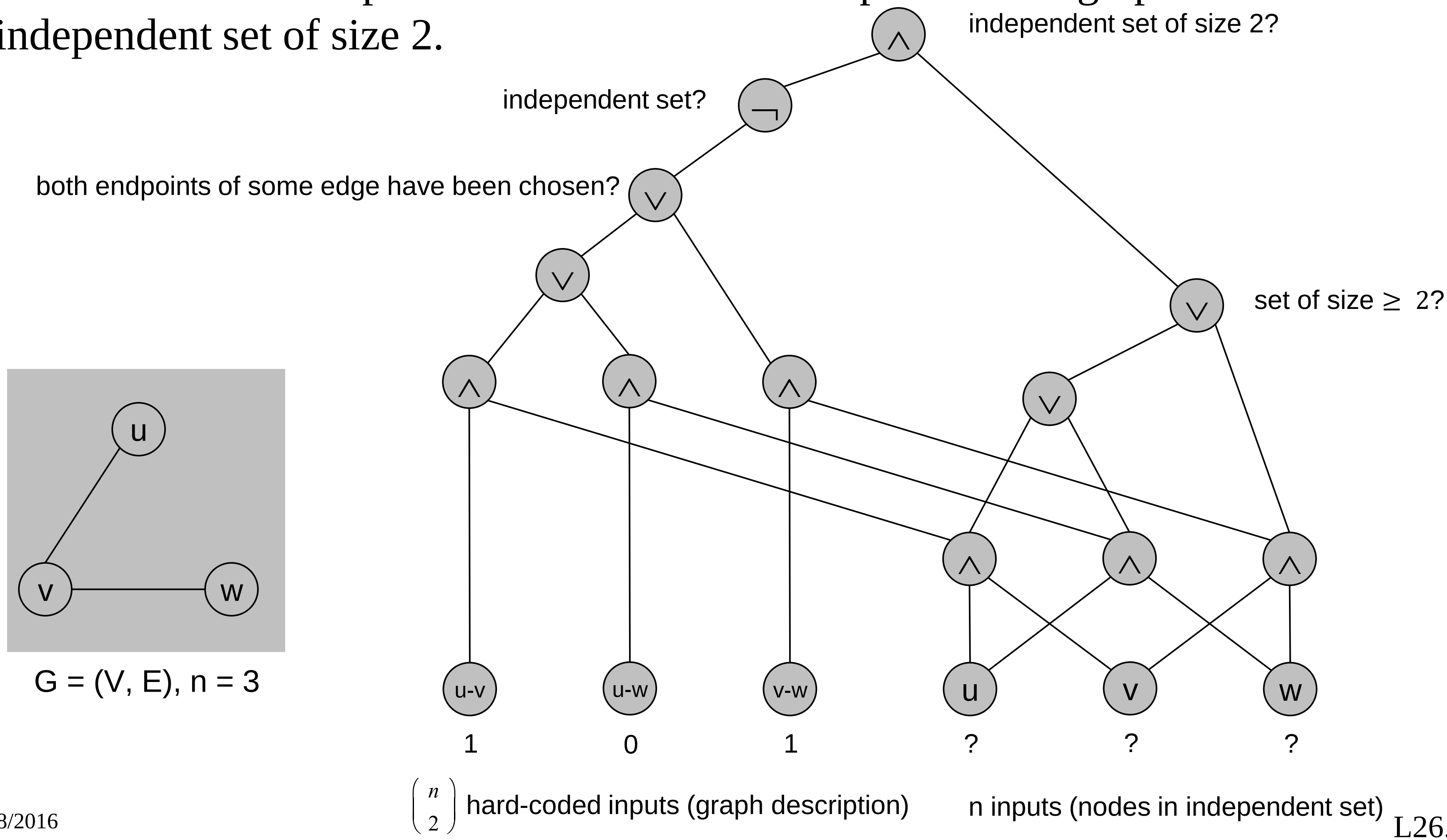
Correctness: Circuit C is satisfiable iff V(w, c) accepts.

Idea #1 (on board): Algorithms are circuits

- Every circuit is an "algorithm" with
  - Fixed length input and
  - Fixed running time
- Every algorithm with fixed length input m, and upper bound t on running time can be converted into a circuit with $O\big(t(m+t)\big)$ gates
  - Space usage is at most $m+t$
  - Imagine a $(m+t) \times t$ table where column i is the state of the algorithms memory at time i
  - Circuit just needs to compute state at time $i+1$ from state at time i.

Example

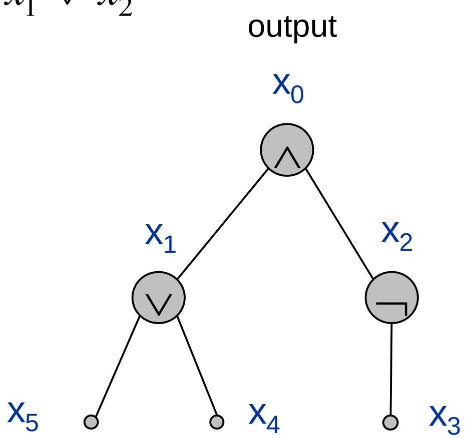
A circuit C whose inputs can be set so that C outputs true iff graph G has an independent set of size 2.



3SAT is NP-complete

Proof. Suffices to show that CIRCUIT-SAT $\leq_P$ 3SAT since 3SAT is in NP.

- Let C be any circuit.
- Create a 3-SAT variable x_i for each circuit element i.
- Make circuit compute correct values at each node:
  - $x_2 = \neg x_3 \Rightarrow$ add 2 clauses: $x_2 \vee x_3$, $\overline{x_2} \vee \overline{x_3}$
  - $x_1 = x_4 \vee x_5 \Rightarrow$ add 3 clauses: $x_1 \vee \overline{x_4}$, $x_1 \vee \overline{x_5}$, $\overline{x_1} \vee x_4 \vee x_5$
  - $x_0 = x_1 \wedge x_2 \Rightarrow$ add 3 clauses: $\overline{x_0} \vee x_1$, $\overline{x_0} \vee x_2$, $x_0 \vee \overline{x_1} \vee \overline{x_2}$
- Hard-coded input values and output value.
  - $x_5 = 0 \Rightarrow$ add 1 clause: $\overline{x_5}$
  - $x_0 = 1 \Rightarrow$ add 1 clause: x_0



- Final step: turn clauses of length < 3 into clauses of length exactly 3.

CS 464

Idea #2 (on board): Verifying is easier than computing

- Given a circuit C with n input wires and m gates, the reduction constructs a 3CNF formula Φ with $O(n+m)$ variables and $O(m)$ clauses
  - One variable Φ for each wire in C
  - Given a satisfying assignment a for C, get a satisfying assignment to Φ:
    - Set variable x_e to the value on wire e when C is executed on a
- Notice the power of nondeterminism
  - Checking that each gate is executed correctly only requires looking at 3 wires at a time