CS 154

Cook-Levin Theorem, NP-Complete Problems

Is SAT solvable in O(n) time on a multitape TM?

Logic circuits of 6n gates for SAT?

If yes, then not only is P=NP, but there would be a "dream machine" that could crank out short proofs of theorems, quickly optimize all aspects of life… recognizing quality work is all you need to produce

THIS IS AN OPEN QUESTION!

Polynomial Time Reducibility

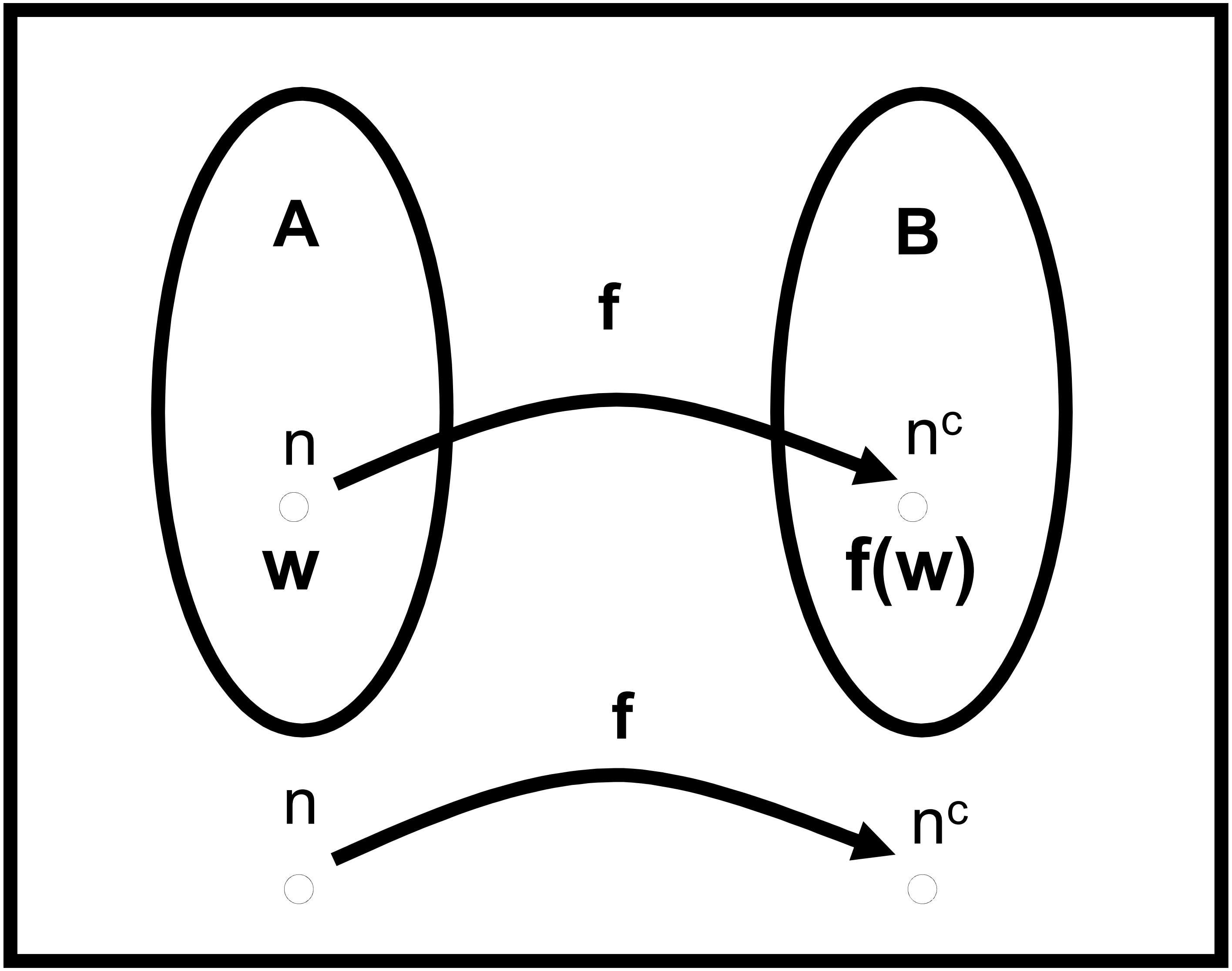
$f : \Sigma^* \to \Sigma^*$ is a polynomial time computable function
if there is a poly-time Turing machine M that on every input w, halts with just f(w) on its tape

Language A is poly-time reducible to language B, written as $A \leq_P B$, if there is a poly-time computable $f : \Sigma^* \to \Sigma^*$ so that:

$$w \in A \Leftrightarrow f(w) \in B$$

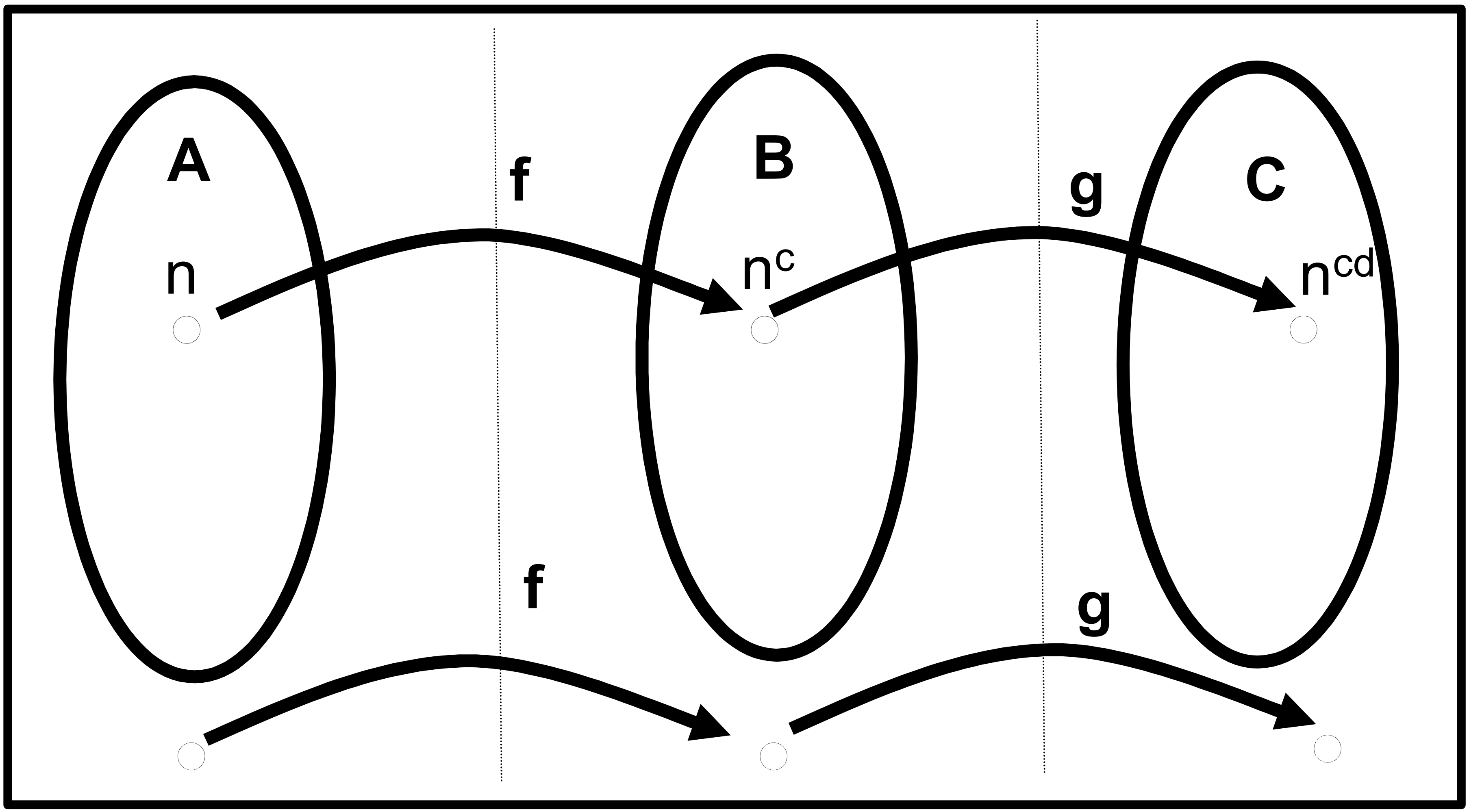
f is a polynomial time reduction from A to B

Note there is a k such that for all w, $|f(w)| \leq |w|^k$



f converts any string w into a string f(w) such that
$w \in A \iff f(w) \in B$

Theorem: If $A \leq_P B$ and $B \leq_P C$, then $A \leq_P C$



Theorem: If $A \leq_P B$ and $B \in P$, then $A \in P$

Proof: **Let M_B be a poly-time TM that decides B.**
Let f be a poly-time reduction from A to B.

We build a machine M_A that decides A as follows:

M_A = On input w,

1. Compute f(w)

2. Run M_B on f(w), output its answer

$$w \in A \Leftrightarrow f(w) \in B$$

Theorem: If $A \leq_P B$ and $B \in NP$, then $A \in NP$

Proof: **Analogous…**

Theorem: If A $\leq_P$ B and B $\in$ P, then A $\in$ P

Theorem: If A $\leq_P$ B and B $\in$ NP, then A $\in$ NP

Corollary: If A $\leq_P$ B and A $\notin$ P, then B $\notin$ P

Definition: A language B is NP-complete if:

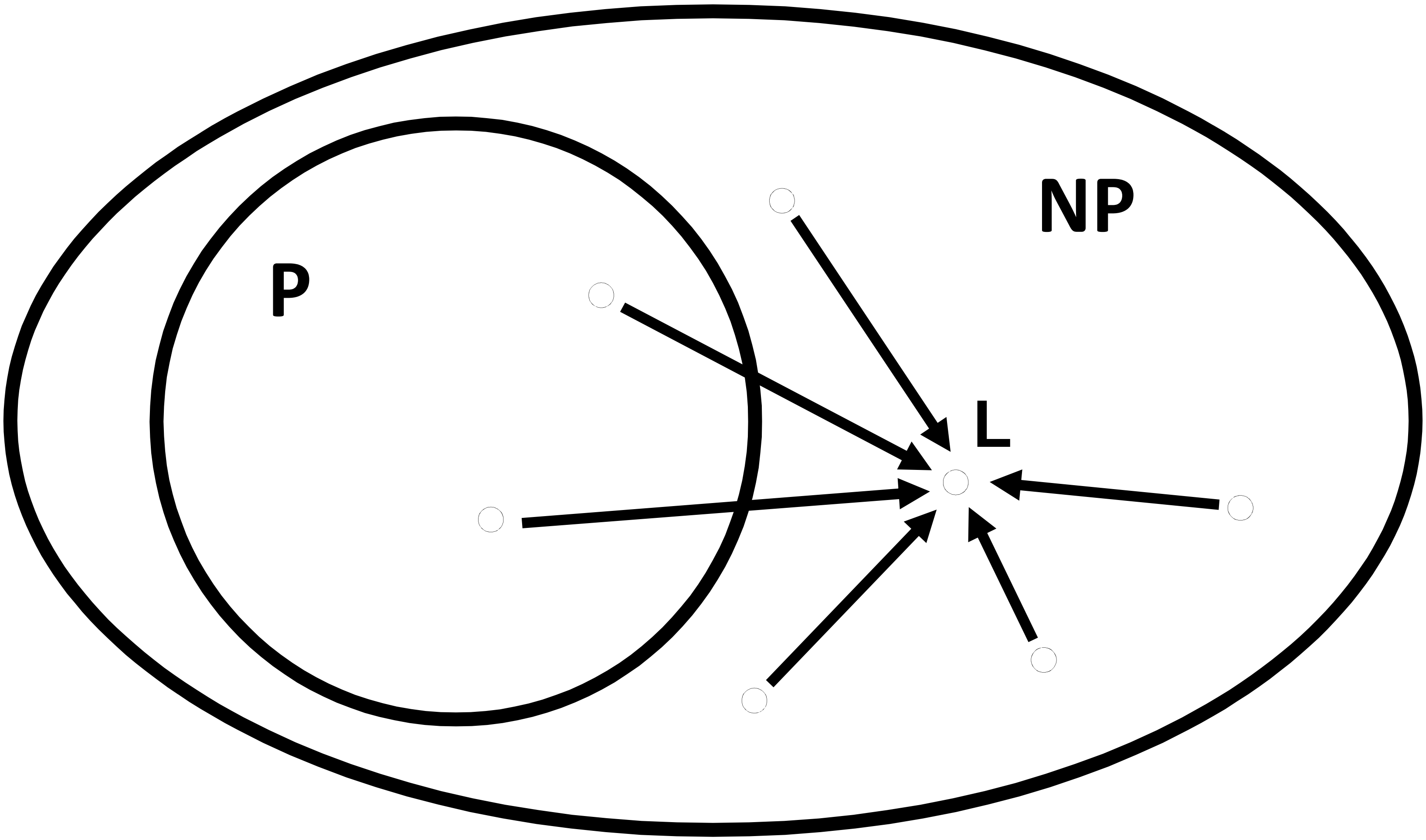
1. B $\in$ NP

2. Every A in NP is poly-time reducible to B
That is, $A \leq_P B$
When this is true, we say "B is NP-hard"

On homework, you showed
A language L is recognizable iff $L \leq_m A_{TM}$

A_{TM} is "*complete for recognizable languages*":
A_{TM} is recognizable, and for all recognizable L, $L \leq_m A_{TM}$

Suppose L is NP-Complete…



If $L \in P$, then $P = NP$!

If $L \notin P$, then $P \neq NP$!

Suppose L is NP-Complete…

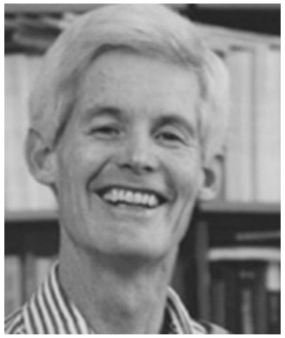
Then assuming the conjecture P $\neq$ NP,

L is not decidable in n^k time, for *every* k

There are thousands of NP-complete problems!

Your favorite topic certainly has an NP-complete problem somewhere in it

Even the other sciences are not safe: biology, chemistry, physics have NP-complete problems too!



The Cook-Levin Theorem:
SAT and 3SAT are NP-complete

1. **3SAT $\in$ NP**
   A satisfying assignment is a "proof" that a 3cnf formula is satisfiable

2. **3SAT is NP-hard**
   Every language in NP can be polynomial-time reduced to 3SAT (complex logical formula)

Corollary: 3SAT $\in$ P if and only if P = NP

Theorem (Cook-Levin): 3SAT is NP-complete

Proof Idea:

(1) 3SAT $\in$ NP (done)

(2) Every language A in NP is polynomial time reducible to 3SAT (this is the challenge)

We give a poly-time reduction from A to SAT

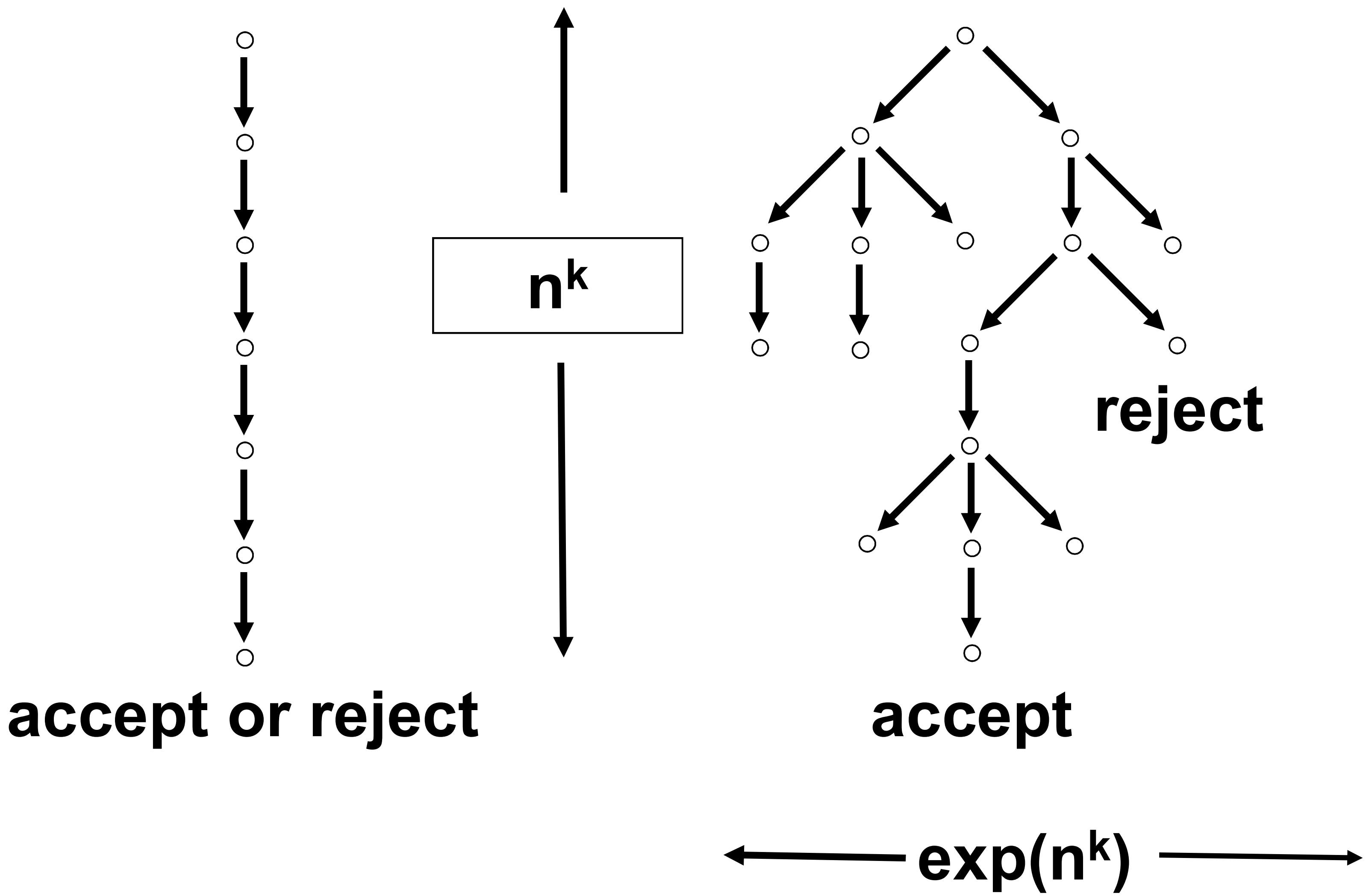
The reduction converts a string w into a 3cnf formula ϕ such that w $\in$ A iff ϕ $\in$ 3SAT

For any A $\in$ NP, let N be a nondeterministic TM deciding A in n^k time

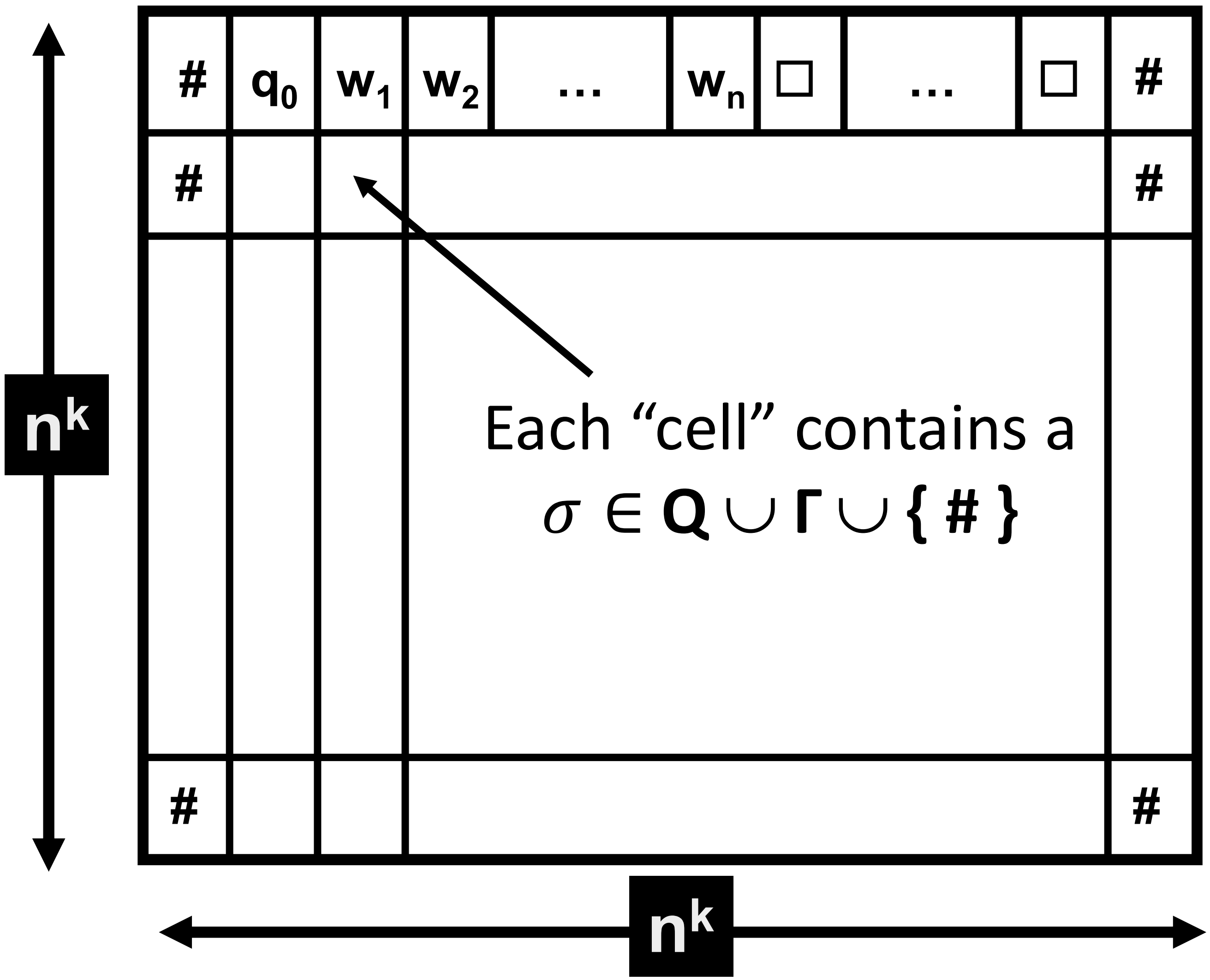
ϕ will simulate N on w

Deterministic Computation

Nondeterministic Computation



Let $L(N) \in \mathbf{NTIME}(n^k)$. A tableau for N on w is an $n^k \times n^k$ matrix whose rows are the configurations of *some* possible computation history of N on w



A tableau is accepting if the last row of the tableau is an accepting configuration

N accepts w if and only if
there is an accepting tableau for N on w

Given w, we'll construct a 3cnf formula ϕ with $O(|w|^{2k})$ clauses, describing logical constraints that any accepting tableau for N on w must satisfy

The 3cnf formula ϕ will be satisfiable *if and only if*
there is an accepting tableau for N on w

Variables of formula ϕ will *encode* a tableau

Let $C = Q \cup \Gamma \cup \{ \# \}$

Each cell of a tableau contains a symbol from C

cell[i,j] = symbol in the cell at row i and column j
= the jth symbol in the ith configuration

For every i and j ($1 \leq i, j \leq n^k$) and for every $s \in C$ we make a Boolean variable $x_{i,j,s}$ in ϕ

Total number of variables = $|C|n^{2k}$, which is $O(n^{2k})$

The $x_{i,j,s}$ variables represent the cells of a tableau

We will enforce the condition: for all i, j, s,

$$x_{i,j,s} = 1 \Leftrightarrow \text{cell}[i,j] = s$$

Idea: Make ϕ so that every *satisfying assignment* to the variables $x_{i,j,s}$ corresponds to an *accepting tableau* for N on w (an assignment to all cell[i,j]'s of the tableau)

The formula ϕ will be the AND of four CNF formulas:

$$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$$

ϕ_{cell} : for all i, j, there is a *unique* $s \in C$ with $x_{i,j,s} = 1$

ϕ_{start} : the first row of the table equals the *start* configuration of N on w

ϕ_{accept} : the last row of the table has an accept state

ϕ_{move} : every row is a configuration that yields the configuration on the next row

ϕ_{start} **: the first row of the table equals the *start* configuration of N on w**

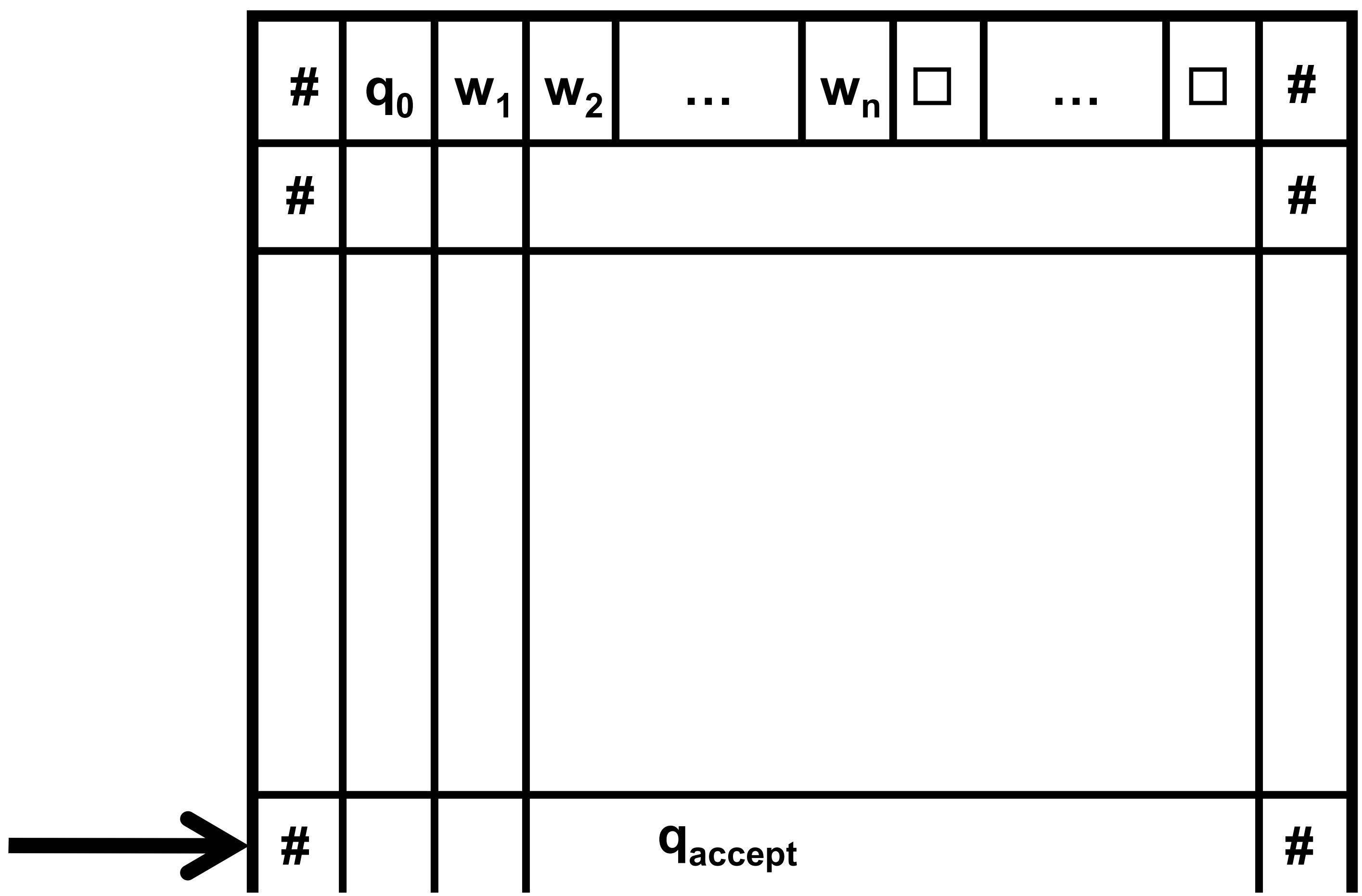
$$\phi_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \cdots \wedge x_{1,n+2,w_n} \wedge x_{1,n+3,\square} \wedge \cdots \wedge x_{1,n^k-1,\square} \wedge x_{1,n^k,\#}$$

#	q_0	w_1	w_2	…	w_n	$\square$	…	$\square$	#
#									#

$O(n^k)$ clauses

ϕ_{accept} : the last row of the table has an accept state

$$\phi_{accept} = \bigvee_{1 \le j \le n^k} x_{n^k, j, q_{accept}}$$



ϕ_{accept} : the last row of the table has an accept state

$$\phi_{accept} = \bigvee_{1 \leq j \leq n^k} x_{n^k, j, q_{accept}}$$

How can we convert ϕ_{accept} into a 3-cnf formula?

The clause $(a_1 \vee a_2 \vee \ldots \vee a_t)$ is equivalent to

$(a_1 \vee a_2 \vee z_1) \wedge (\neg z_1 \vee a_3 \vee z_2) \wedge \ldots \wedge (\neg z_{t-3} \vee a_{t-1} \vee a_t)$

where z_i are new variables.

This produces O(t) new 3cnf clauses.

$O(n^k)$ clauses

ϕ_{cell} : for all i, j, there is a unique $s \in C$ with $x_{i,j,s} = 1$

$$\phi_{cell} = \bigwedge_{1 \le i,\, j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right]$$

for all i,j

at least one $x_{i,j,s}$ is set to 1

at most one $x_{i,j,s}$ is set to 1

$O(n^{2k})$ clauses

ϕ_{move} **: every row is a configuration that yields the configuration on the next row**

Key Question: If one row yields the next row, how many cells can be different between the two rows?

Answer: AT MOST THREE CELLS!

#	b	a	a	q_1	b	c	b	#
#	b	a	q_2	a	c	c	b	#

ϕ_{move} **: every row is a configuration that yields the configuration on the next row**

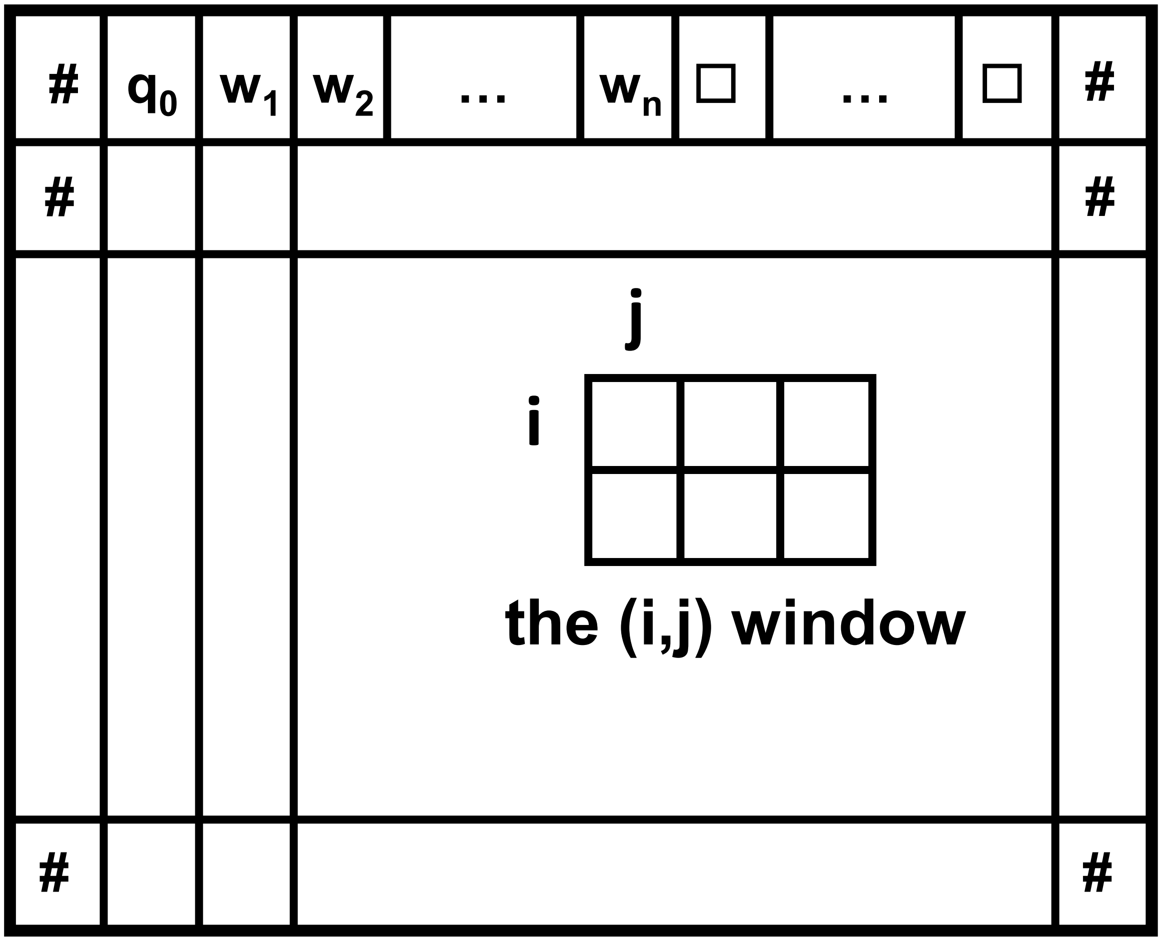
Key Question: If one row yields the next row, how many cells can be different between the two rows?

Answer: AT MOST THREE CELLS!

#	b	a	a	q_1	b	c	b	#
#	b	a	q_2	a	c	c	b	#

ϕ_{move} : every row is a configuration that yields the configuration on the next row

Idea: check that every 2×3 "window" of cells is legal (consistent with the transition function of N)



If $\delta(q_1,a) = \{(q_1,b,R)\}$ and $\delta(q_1,b) = \{(q_2,c,L), (q_2,a,R)\}$ which of the following windows are legal?

a	q_1	b
q_2	a	c

a	q_1	b
q_1	a	a

a	a	q_1
a	a	b

#	b	a
#	b	a

a	b	a
a	b	q_2

b	q_1	b
q_2	b	q_2

a	b	a
a	a	a

a	q_1	b
a	a	q_2

b	b	b
c	b	b

Key Lemma:

IF Every window of the tableau is legal, and

The 1st row is the start configuration of N on w

THEN for all i = 1,…,n^k – 1, the ith row of the tableau is a configuration which yields the (i+1)th row.

Proof Sketch: (Strong) induction on i.

The 1st row is a configuration. If it *didn't* yield the 2nd row, there's a 2 x 3 "illegal" window on 1st and 2nd rows

Assume rows 1,…,L are all configurations which yield the next row, and assume every window is legal.

If row L+1 did *not* yield row L+2, then there's a 2 x 3 window along those two rows which is "illegal"

The (i, j) window of a tableau is the tuple $(a_1,\ldots, a_6) \in C^6$ such that

	col. j	col. j+1	col. j+2
row i	a_1	a_2	a_3
row i+1	a_4	a_5	a_6

ϕ_{move} : every row is a configuration that legally follows from the previous configuration

$$\phi_{move} = \bigwedge_{\substack{1 \le i \le n^k - 1 \\ 1 \le j \le n^k - 2}} \text{(the (i, j) window is legal)}$$

(the (i, j) window is legal) =

$$\bigvee_{\substack{(a_1, \ldots, a_6) \\ \text{is a legal window}}} (x_{i,j,a_1} \wedge x_{i,j+1,a_2} \wedge x_{i,j+2,a_3} \wedge x_{i+1,j,a_4} \wedge x_{i+1,j+1,a_5} \wedge x_{i+1,j+2,a_6})$$

$$\equiv \bigwedge_{\substack{(a_1, \ldots, a_6) \\ \text{is NOT a legal window}}} (\overline{x}_{i,j,a_1} \vee \overline{x}_{i,j+1,a_2} \vee \overline{x}_{i,j+2,a_3} \vee \overline{x}_{i+1,j,a_4} \vee \overline{x}_{i+1,j+1,a_5} \vee \overline{x}_{i+1,j+2,a_6})$$

$$\phi_{move} = \bigwedge_{1 \le i, j \le n^k} (\text{ the } (i, j) \text{ window is "legal" })$$

the (i, j) window is "legal" =

$$\equiv \bigwedge_{\substack{(a_1, \ldots, a_6) \\ \text{ISN'T "legal"}}} (\overline{x}_{i,j,a_1} \vee \overline{x}_{i,j+1,a_2} \vee \overline{x}_{i,j+2,a_3} \vee \overline{x}_{i+1,j,a_4} \vee \overline{x}_{i+1,j+1,a_5} \vee \overline{x}_{i+1,j+2,a_6})$$

$O(n^{2k})$ clauses

Summary. We wanted to prove:
Every A in NP has a polynomial time reduction to 3SAT

For every A in NP, we know A is decided by some nondeterministic n^k time Turing machine N

We gave a generic method to reduce a string w to a 3CNF formula ϕ of $O(|w|^{2k})$ clauses such that
satisfying assignments to the variables of ϕ
directly correspond to
accepting computation histories of N on w

The formula ϕ is the AND of four 3CNF formulas:

$$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$$

Theorem (Cook-Levin):
SAT and 3SAT are NP-complete

Corollary: SAT $\in$ P if and only if P = NP

Given a favorite problem $\Pi \in$ NP,
how can we prove it is NP-hard?

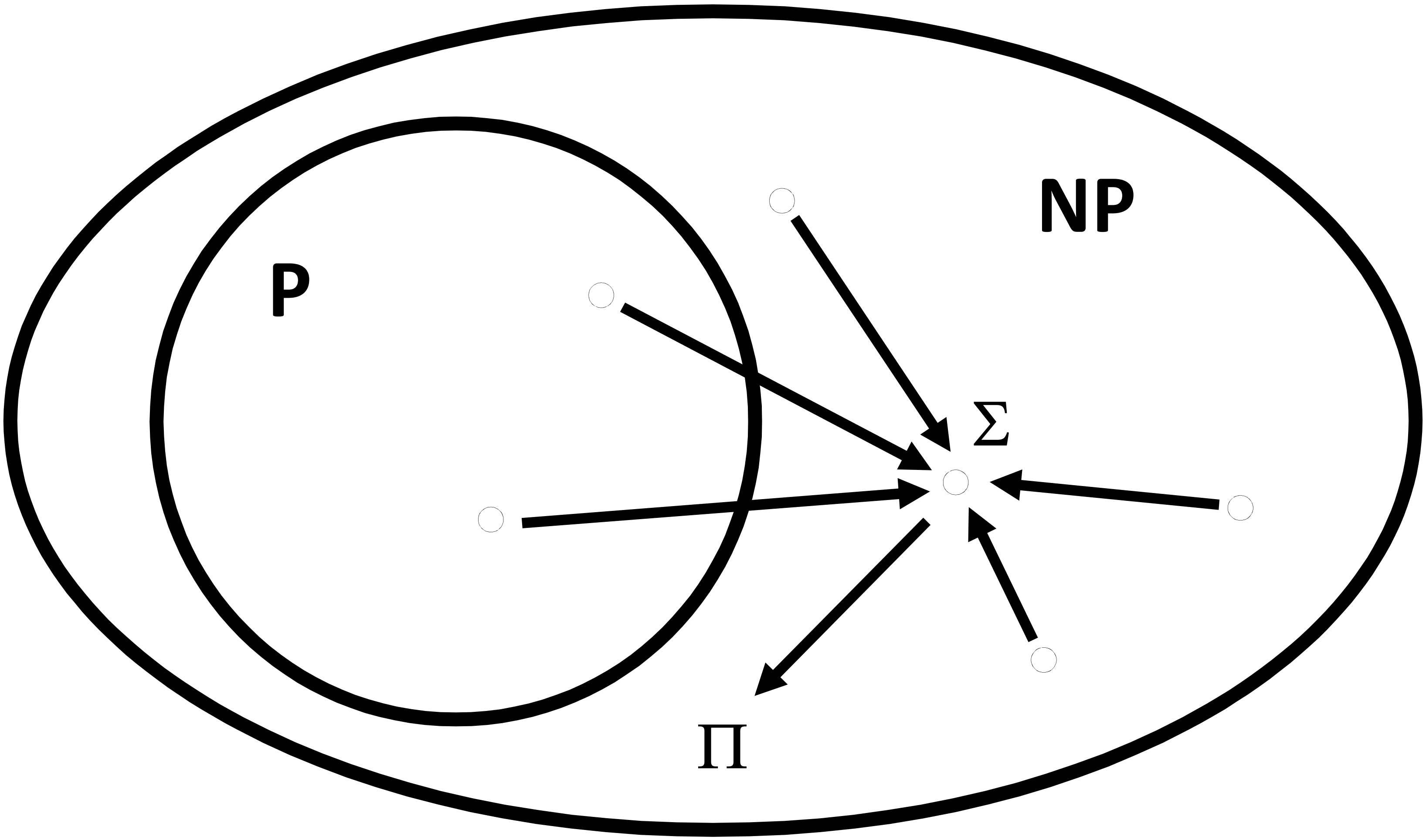
Generic Recipe:

1. **Take a problem Σ that you know to be NP-hard (3-SAT)**
2. **Prove that $\Sigma \leq_P \Pi$**

Then for all A $\in$ NP, A $\leq_P \Sigma$ and $\Sigma \leq_P \Pi$

We conclude that A $\leq_P \Pi$, and Π is NP-hard

Π is NP-Complete



The Clique Problem

Given a graph G and positive k, does G contain a complete subgraph on k nodes?

CLIQUE = { (G,k) | G is an undirected graph with a k-clique }

Theorem (Karp): CLIQUE is NP-complete

Proof Idea: 3SAT $\leq_P$ CLIQUE

Transform a 3-cnf formula ϕ into (G,k) such that

$\phi \in$ 3SAT $\Leftrightarrow$ (G,k) $\in$ CLIQUE

Want transformation that can be done in time that is polynomial in the length of ϕ

How can we encode a *logic* problem as a *graph* problem?

$3SAT \leq_P$ CLIQUE

We transform a 3-cnf formula ϕ into (G,k) such that

$\phi \in$ 3SAT $\Leftrightarrow$ (G,k) $\in$ CLIQUE

Let C_1, C_2, …, C_m be clauses of ϕ. Assign k := m.
Make a graph G with m *groups* of 3 nodes each.

Group *i* corresponds to clause C_i of ϕ

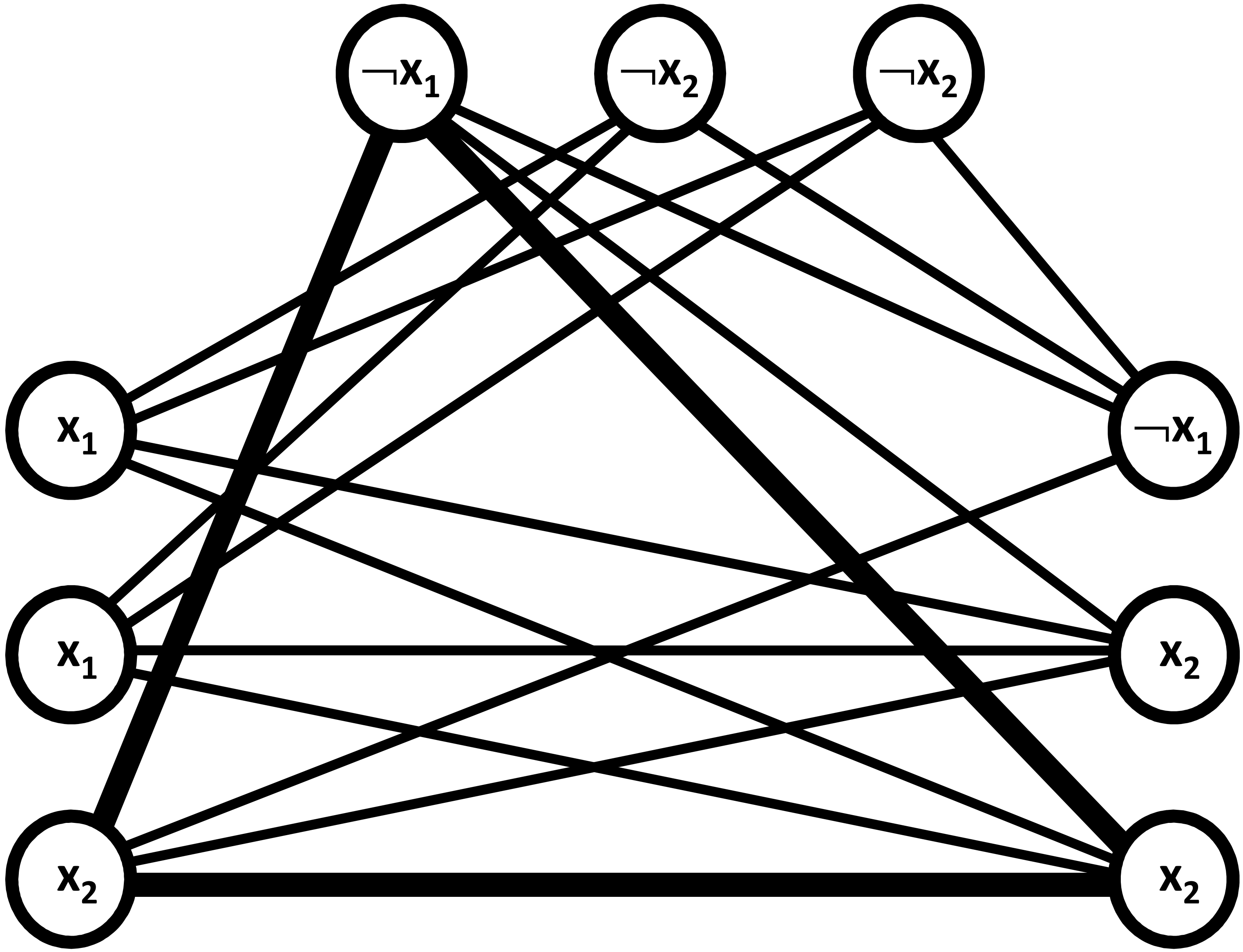
Each node in group *i* is labeled with a literal of C_i

Put edges between all pairs of nodes in different groups,
except pairs of nodes with labels x_i and $\neg x_i$

Put no edges between nodes in the same group

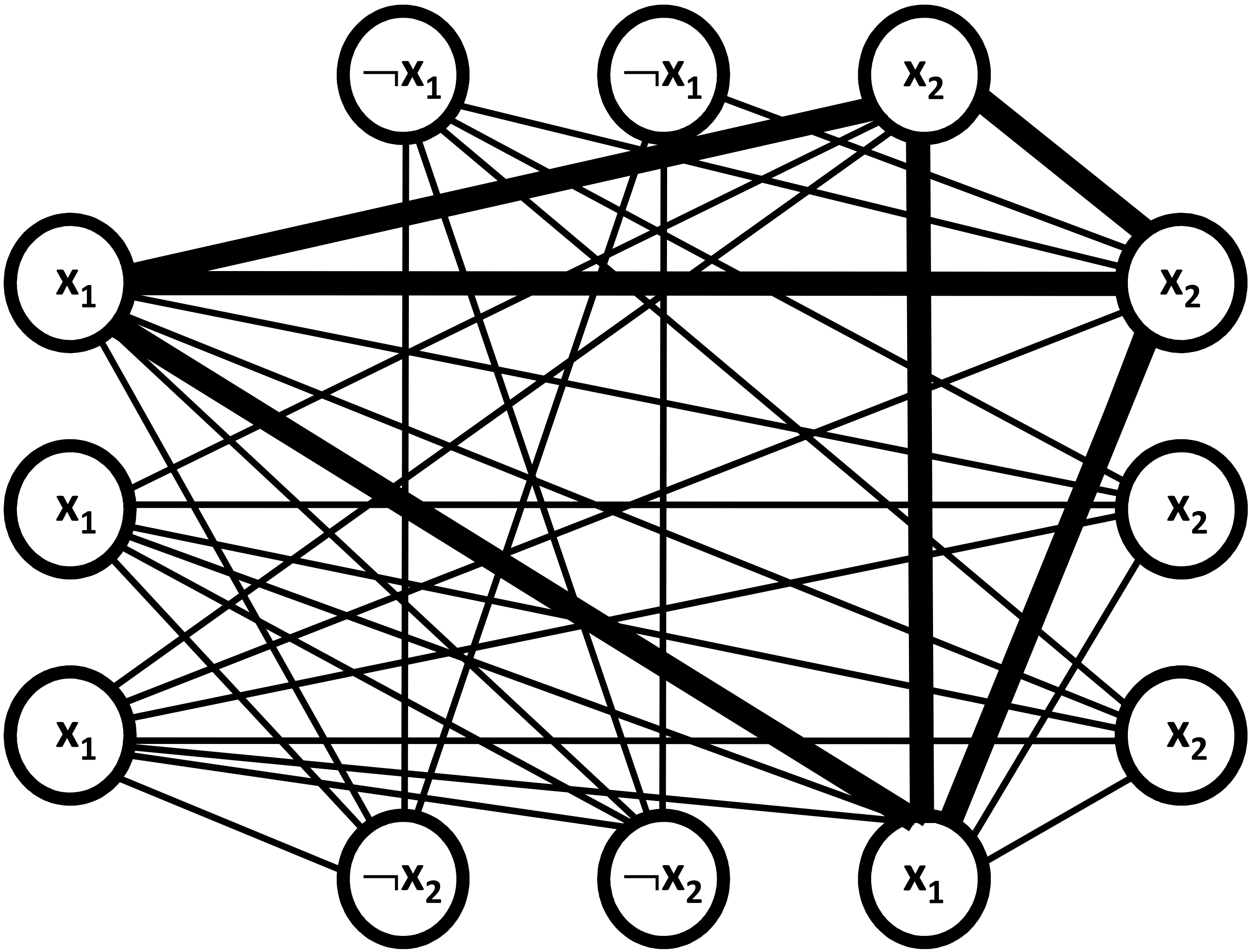
When done putting in all the edges, *erase* the labels

$$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$$



|V| = 9 k = 3

$$(x_1 \lor x_1 \lor x_1) \land (\neg x_1 \lor \neg x_1 \lor x_2) \land$$
$$(x_2 \lor x_2 \lor x_2) \land (\neg x_2 \lor \neg x_2 \lor x_1)$$



Claim: $\phi \in$ 3SAT $\Leftrightarrow$ (G,m) $\in$ CLIQUE

Claim: If $\phi \in$ 3SAT then (G,m) $\in$ CLIQUE

Proof: Given a SAT assignment A of ϕ, for every clause C there is at least one literal in C that's set true by A

For each clause C, let v_C be a vertex from group C whose label is a literal that is set true by A

Claim: S = $\{v_C : C \in \phi\}$ is an m-clique

Proof: Let $v_C, v_{C'}$ be in S. Suppose $(v_C, v_{C'}) \notin$ E.

Then v_C and $v_{C'}$ must label *inconsistent* literals, call them x and $\neg$x

But assignment A cannot satisfy both x and $\neg$x

Therefore $(v_C, v_{C'}) \in$ E, for all $v_C, v_{C'} \in$ S.

Hence S is an m-clique, and (G,m) $\in$ CLIQUE

Claim: $\phi \in$ 3SAT $\Leftrightarrow$ (G,m) $\in$ CLIQUE

Claim: If (G,m) $\in$ CLIQUE then $\phi \in$ 3SAT
Proof: Let S be an m-clique of G.
We construct a satisfying assignment A of ϕ.

Claim: S contains *exactly one node* from each group.

Now for each variable x of ϕ, make assignment A:
Assign x to 1 $\Leftrightarrow$ There is a vertex v $\in$ S with label x

For all i = 1,…,m, at least one vertex from group i is in S.
Therefore, for all i = 1,…,m
A satisfies at least one literal in the ith clause of ϕ
Therefore A is a satisfying assignment to ϕ

Independent Set

IS: Given a graph G = (V, E) and integer k,
is there $S \subseteq V$ such that $|S| \geq k$ and

***no* two vertices in S have an edge?**

IS = {(G,k) |

CLIQUE: Given G = (V, E) and integer k,
is there $S \subseteq V$ such that $|S| \geq k$

and every pair of vertices in S have an edge?

CLIQUE $\leq_P$ IS:

Given G = (V, E), output G' = (V, E') where

E' = {(u,v) | (u,v) $\notin$ E}.

(G, k) $\in$ CLIQUE iff (G', k) $\in$ IS

Reading Assignment

Read Luca Trevisan's notes for an alternative proof of the Cook-Levin Theorem!

Sketch:

1. **Define CIRCUIT-SAT:** ***Given a logical circuit C(y), is there an input a such that C(a)=1?***
2. **Show that CIRCUIT-SAT is NP-hard:**
   The n^k x n^k tableau for N on w can be simulated using a logical circuit of $O(n^{2k})$ gates
3. **Reduce CIRCUIT-SAT to 3SAT in polytime**
4. **Conclude 3SAT is also NP-hard**