# CS 154

## NP-Completeness and Cook's Theorem

$$NP = \bigcup_{k \in N} NTIME(n^k)$$

Theorem: $L \in NP \Leftrightarrow$ there exists a poly-time Turing machine V and k s.t. we can define L as:

$L = \{ x \mid \exists y\ [|y| \leq |x|^k$ and $V(x,y)$ accepts $] \}$

3SAT = $\{ \phi \mid \exists y$ such that y is a satisfying assignment to $\phi$ and $\phi$ is in 3cnf $\}$

SAT = $\{ \phi \mid \exists y$ such that y is a satisfying assignment to $\phi \}$

**A language is in NP if and only if there are polynomial-length proofs for membership in the language**

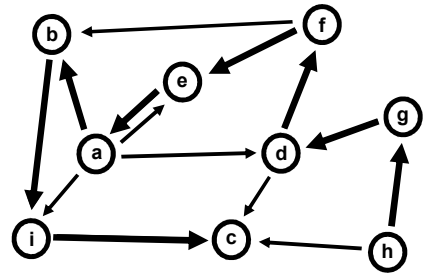
SAT is in NP because a satisfying assignment is a polynomial-length proof that a formula is satisfiable

When $\phi \in$ SAT, I can prove that fact to you with a short proof you can quickly verify

NP = All the problems for which the solvable instances have short answers, and once you *have* the answer, you can efficiently verify it

$L \in NP$ iff
$L = \{x \mid \exists$ Answer A of $|x|^k$ bits) [Answer A to x is correct]$\}$

**The Hamiltonian Path Problem**




A Hamiltonian path traverses through each node exactly once

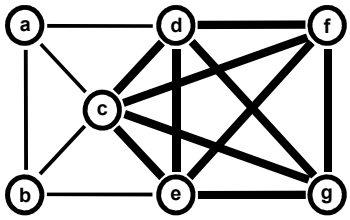
Assume a reasonable encoding of graphs (example: the adjacency matrix is reasonable)

HAMPATH = { (G,s,t) | G is a directed graph with a Hamiltonian path from s to t }

Theorem: HAMPATH $\in$ NP

The Hamiltonian path is a proof that (G,s,t) is a member of HAMPATH

## The k-Clique Problem




k-clique = complete subgraph on k nodes

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

Theorem: CLIQUE $\in$ NP

A k-clique itself is a proof that (G, k) is in CLIQUE

Given the k-clique, you can verify that all {k choose 2} edges are present

A language is in NP if and only if there exist "polynomial-length proofs" for membership in the language

**P** = the problems that can be efficiently solved

**NP** = the problems where *proposed solutions to the solvable instances can be efficiently verified*

# P = NP?

*"Can Problem Solving Be Efficiently Automated?"*

A Clay Institute Millennium Problem

## If P = NP…

Mathematicians may be out of a job

Cryptography as we know it would probably be impossible

In principle, every aspect of daily life could be efficiently and globally optimized

# Conjecture: P $\neq$ NP

## Poly-Time Reducibility

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

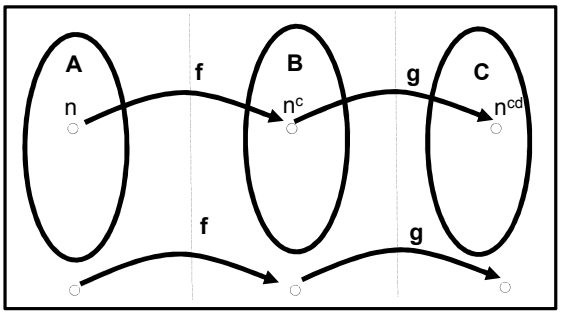
Language A is polynomial time reducible to language B, written $A \leq_P B$, if there is a poly-time computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that:

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

f is a polynomial time reduction from A to B

Note: $|f(w)| \leq |w|^k$ for some constant k

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

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

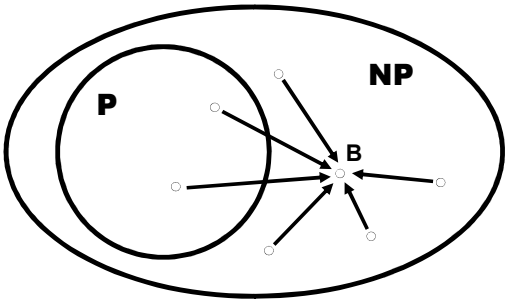
Theorem: If $A \leq_P B$ and $B \in P$, then $A \in P$

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, $A \leq_P B$
   (B is NP-hard)

## Suppose B is NP-Complete…




If B is NP-Complete and $B \in P$, then P = NP!

## Suppose B is NP-Complete…

Then assuming the conjecture $P \neq NP$,

B is not solvable in $O(n^k)$ time, for *any* k

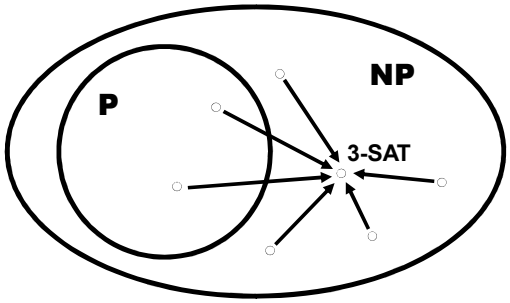
**Theorem (Cook-Levin):**
**SAT and 3-SAT are NP-complete**

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 polytime reduced to 3SAT (complex logical formula)**

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

## 3-SAT is NP-Complete




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

**Proof Idea:**

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

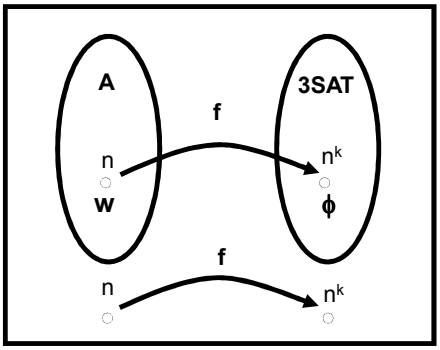
**(2) Every language A in NP is polynomial time reducible to 3SAT (this is the challenge)**

**We give a poly-time reduction from A to SAT**

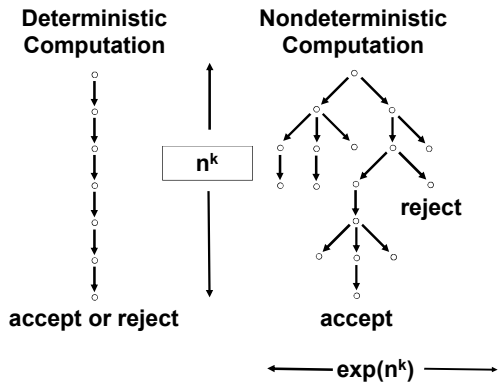
**The reduction turns a string w into a 3cnf formula $\phi$ such that $w \in A$ iff $\phi \in$ 3SAT.**

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

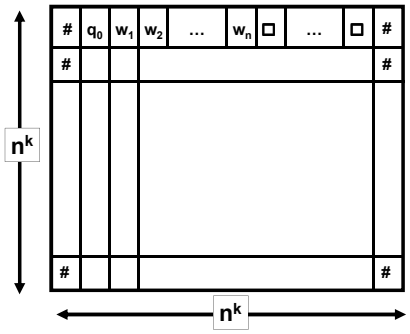
**$\phi$ will simulate the NP machine N for A on w.**




**f turns any string w into a 3-cnf formula $\phi$ such that $w \in A \Leftrightarrow \phi$ is satisfiable.**
**$\phi$ will simulate an NP machine N on w, where A = L(N)**




**A tableau for N on w is an $n^k \times n^k$ table whose rows are the configurations of *some* possible computation of N on input w ($L(N) \in NTIME(n^k)$)**



A tableau is accepting if any 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, our Boolean 3cnf formula $\phi$ will describe all the logical constraints that any accepting tableau for N on w must satisfy.

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

## Variables of our formula $\phi$

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

Each of the $(n^k)^2$ entries of a tableau is a cell

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

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

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

These $x_{i,j,s}$ are the variables of $\phi$ and represent the contents of the cells

We will have: $x_{i,j,s} = 1 \Leftrightarrow$ cell[i,j] = s

$$x_{i,j,s} = 1$$

**means**

**cell[ i, j ] = s**

We now design $\phi$ so that a satisfying assignment to the variables $x_{i,j,s}$ corresponds to an accepting tableau for N on w (an assignment to the cell[i,j])

The formula $\phi$ will be the AND of four CNF formulas:
$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$

$\phi_{cell}$ : for all i, j, *exactly one* $s \in C$ has $x_{i,j,s} = 1$

$\phi_{start}$ : the first row of the table is the *start* configuration of N on w

$\phi_{accept}$ : an accepting configuration is the last row of the table

$\phi_{move}$ : every row is a configuration that legally follows from the previous configuration

$\phi_{cell}$ : for all i, j, *exactly one* $s \in C$ has $x_{i,j,s} = 1$

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

for all i,j — at least one variable is set to 1 — at most one variable is set to 1

$\phi_{start}$ : the first row of the tableau is the *start* configuration of N on w

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

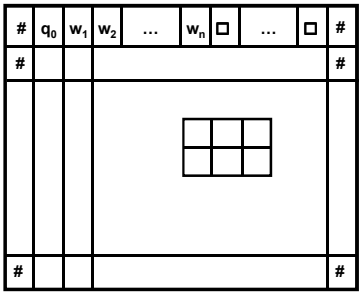
| # | $q_0$ | $w_1$ | $w_2$ | … | $w_n$ | $\square$ | … | $\square$ | # |
|---|---|---|---|---|---|---|---|---|---|
| # | | | | | | | | | # |
| | | | | | | | | | |

$\phi_{accept}$ : an accepting configuration is the last row of the table

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

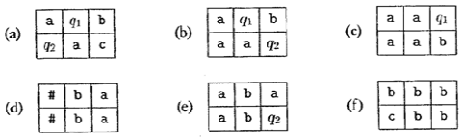
$\phi_{move}$ : every row is a configuration that legally follows from the previous configuration

This formula checks that every $2 \times 3$ "window" of cells is legal (it obeys the transition function)

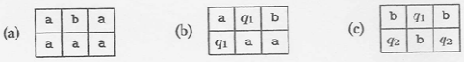



Example $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
Suppose $a, b, c \in \Gamma$, $q_1, q_2 \in Q$ and
$\delta(q_1,a) = \{ (q_1, b, R)\}$
$\delta(q_1, b) = \{ (q_2, c, L), (q_2, a, R) \}$

Legal = Consistent with N's transition function




Illegal = Inconsistent with N's transition function




Lemma:
IF
- the top row of the tableau is the start configuration, and
- every window of the tableau is legal,

THEN
each row of the table (as a configuration) yields the next row on the table (as a configuration)

Proof: Very similar to the PCP's undecidability.
If there is no state in the window, then this is easy: the top of the window equals the bottom
If there is a state, then the window has to obey what the transition function can do in one step.

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

| | col. j | col. j+1 | col. j+2 |
|---|---|---|---|
| row i | (i, j) $a_1$ | (i, j+1) $a_2$ | (i, j+2) $a_3$ |
| row i+1 | (i+1, j) $a_4$ | (i+1, j+1) $a_5$ | (i+1, j+2) $a_6$ |

$\phi_{move}$ : 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) =

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

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

## 3-SAT?

**We had some crazy clauses in there… how do we convert the whole thing into a 3-cnf formula?**

**Everything was an AND of ORs (a CNF)**
**We just need to make those ORs small**

$$(a_1 \vee a_2 \vee \ldots \vee a_t) =$$
$$(a_1 \vee a_2 \vee z_1) \wedge (\neg z_1 \vee a_3 \vee z_2) \wedge (\neg z_2 \vee a_4 \vee z_3) \ldots$$

What's the total length of $\phi$?

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

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

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

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

# $O(n^k)$ clauses

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

$$(a_1 \vee a_2 \vee \ldots \vee a_t) =$$
$$(a_1 \vee a_2 \vee z_1) \wedge (\neg z_1 \vee a_3 \vee z_2) \wedge (\neg z_2 \vee a_4 \vee z_3) \ldots$$

**yields O(t) new 3cnf clauses**

# $O(n^k)$ clauses

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

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

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

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

Summary: Made $\phi$ so that a satisfying assignment to the variables $x_{i,j,s}$ corresponds to an accepting tableau for N on w (an assignment to the cell[i,j])

The formula $\phi$ is the AND of four CNF formulas:

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

$\phi_{cell}$ : for all i, j, *exactly one* $s \in C$ has $x_{i,j,s} = 1$

$\phi_{start}$ : the first row of the table is the *start* configuration of N on w

$\phi_{accept}$ : an accepting configuration is the last row of the table

$\phi_{move}$ : every row is a configuration that legally follows from the previous configuration

## Reading Assignment

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

Sketch: For every *deterministic* $n^k$ time V(x,y),

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 $n^k \times n^k$ tableau for V can be computed with a logical circuit of $O(n^{2k})$ gates
3. Reduce CIRCUIT-SAT to 3-SAT in polytime
4. Conclude 3-SAT is also NP-hard

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

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

There are thousands of NP-complete problems

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

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

Is 3SAT solvable in O(n) time on a multitape TM?

If yes, then not only is P=NP true, but we would have a "dream machine" that could crank out short proofs of theorems, extreme optimization in all aspects of life…

**THIS IS AN OPEN QUESTION!**

# Next Episode:

**More NP-Completeness**