CS 154

NP-Completeness and the Cook-Levin Theorem

Feedback	Number of students
HW is too hard	10
HW is too easy	0
Pace is too fast	4
Pace/hw is just right	10
Likes Ryan/slides/lecs	53
Hates streaming/comm.	4
Likes streaming/comm.	3
Likes TAs	10
More feedback from TAs	7
More Office Hours	4
Likes examples	10
Want review sessions	2
Too slow / too much like 103	4
Not enough 103 (wants PDAs)	1
Hates “boxes”	1
Hates exams	2
I dislike that I don’t have any criticism	3

$$\mathbf{P} = \bigcup_{k \in \mathbf{N}} \mathrm{TIME}(n^k)$$

Definition: NTIME(t(n)) =

{ L | L is decided by a O(t(n)) time nondeterministic Turing machine }

TIME(t(n)) $\subseteq$ NTIME(t(n))

Is TIME(t(n)) = NTIME(t(n)) for all t(n)?

THIS IS AN OPEN QUESTION!

Boolean Formulas

A **satisfying assignment** is a setting of the variables that makes the formula true

$$\phi = (\neg x \land y) \lor z$$

x = 1, y = 1, z = 1 is a satisfying assignment for ϕ

$$\neg(x \lor y) \land (z \land \neg x)$$

0 0 1 0

A Boolean formula is **satisfiable** if there exists a true/false setting to the variables that makes the formula true

YES $\quad a \wedge b \wedge c \wedge \neg d$

NO $\quad \neg(x \vee y) \wedge x$

SAT = { ϕ | ϕ is a satisfiable Boolean formula }

A **3cnf-formula** has the form:

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_4 \vee x_2 \vee x_5) \wedge (x_3 \vee \neg x_2 \vee \neg x_1)$$

literals

clauses

Ex: $(x_1 \vee \neg x_2 \vee x_1)$

$(x_3 \vee x_1) \wedge (x_3 \vee \neg x_2 \vee \neg x_1)$

$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_4 \vee x_2 \vee x_1) \vee (x_3 \vee x_1 \vee \neg x_1)$

$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_3 \wedge \neg x_2 \wedge \neg x_1)$

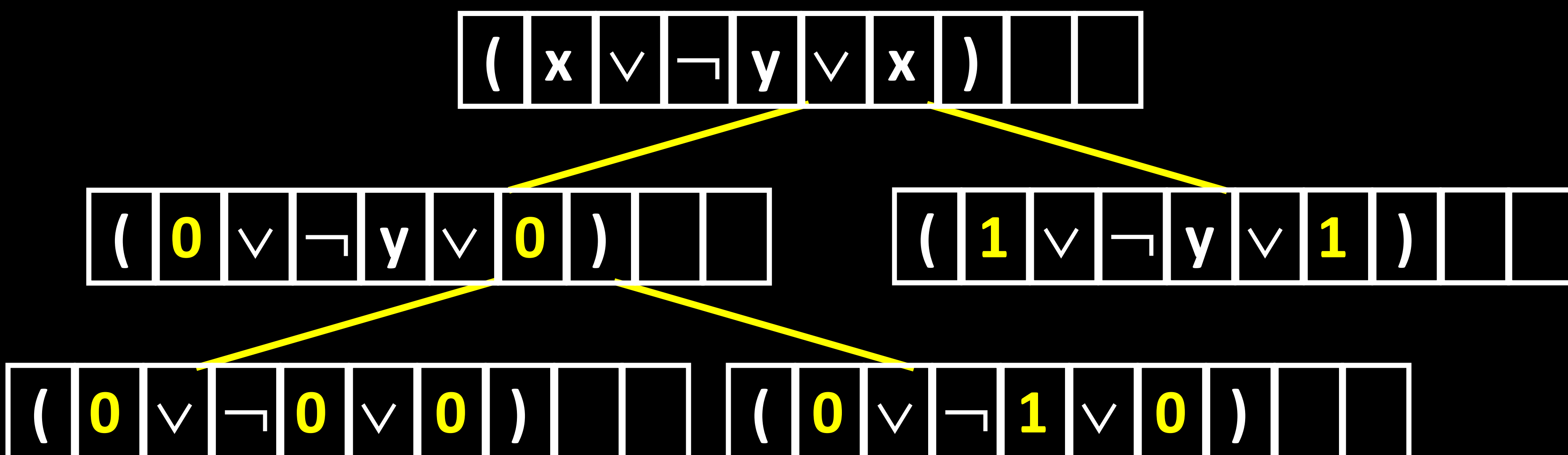
3SAT = { ϕ | ϕ is a satisfiable 3cnf-formula }

3SAT = { ϕ | ϕ is a satisfiable 3cnf-formula }

Theorem: 3SAT $\in$ NTIME(n^2)

On input ϕ:

1. Check if the formula is in 3cnf
2. For each variable v in ϕ, nondeterministically substitute either 0 or 1 in place of v



3. Evaluate the formula and *accept* iff ϕ is true

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

Theorem: $L \in NP \Leftrightarrow$ There is a constant k and polynomial-time TM V such that

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

Proof: (1) If $L = \{ x \mid \exists y\ |y| \leq |x|^k \text{ and } V(x,y) \text{ accepts } \}$ then $L \in NP$

Given x, nondeterministically guess y of length $|x|^k$ then output the answer of V(x,y)

(2) If $L \in NP$ then $L = \{ x \mid \exists y\ |y| \leq |x|^k \text{ and } V(x,y) \text{ accepts } \}$

Let N be a nondeterministic poly-time TM that decides L. Define V(x,y) to accept iff y encodes an accepting computation history of N on x

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

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

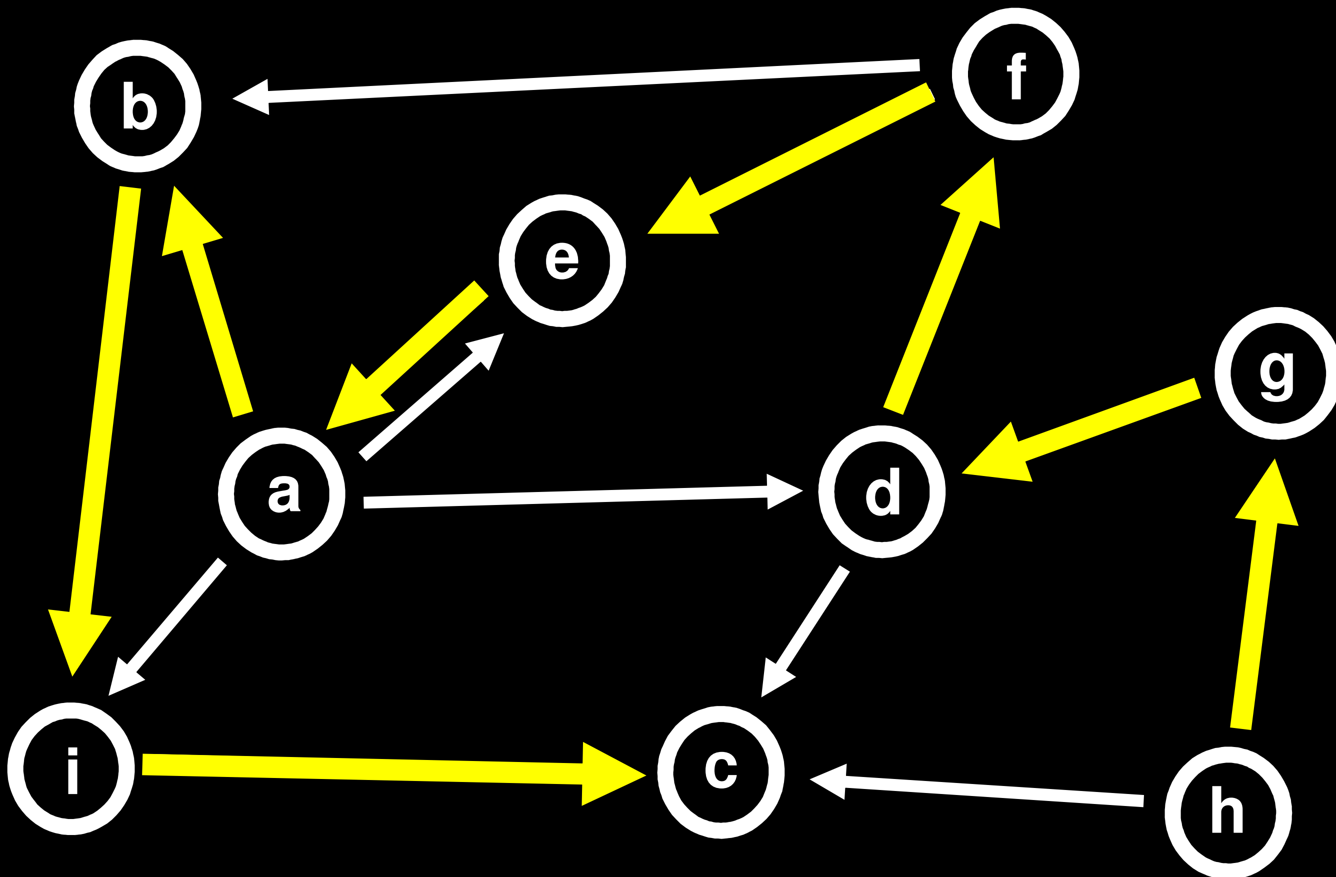
SAT = { ϕ | $\exists$y such that ϕ is a Boolean formula and y is a satisfying assignment to ϕ }

NP = Problems with the property that, once you *have* the answer, it is "easy" to verify the answer

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

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)

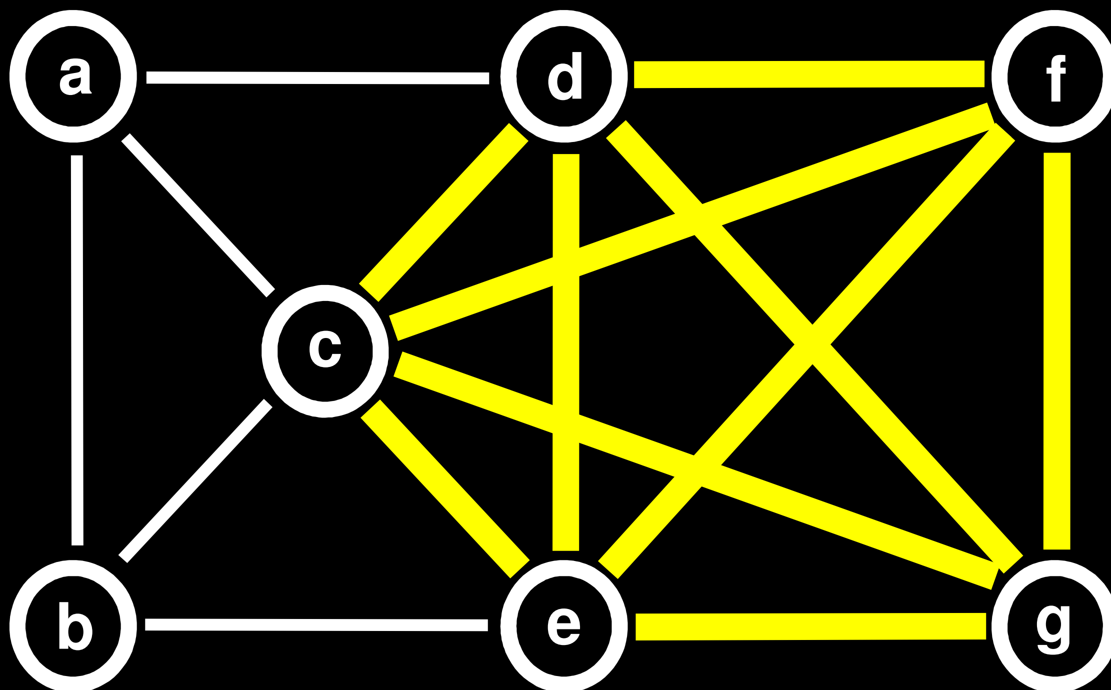
HAMPATH = { (G,s,t) | G is a directed graph with a Hamiltonian path from s to t }

Theorem: HAMPATH $\in$ NP

A Hamiltonian path P in G from s to t is a proof that (G,s,t) is in HAMPATH

Given P (as a permutation on the nodes) can easily check that it is a path through all nodes exactly once

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 in G is a **proof** that (G, k) is in CLIQUE

Given a subset S of k nodes from G, we can efficiently check that all possible edges are present between the nodes in S

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

P = the problems that can be efficiently solved

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

Is P = NP?

can problem solving be automated?

$$$

P = NP?

$$$

If P = NP…

Mathematicians may be out of a job

Cryptography as we know it may be impossible

In principle, every aspect of our lives could be quickly and globally optimized…
… life as we know it would be different

Conjecture: P $\neq$ NP

Polynomial Time Reducibility

$f : \Sigma^* \rightarrow \Sigma^*$ 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

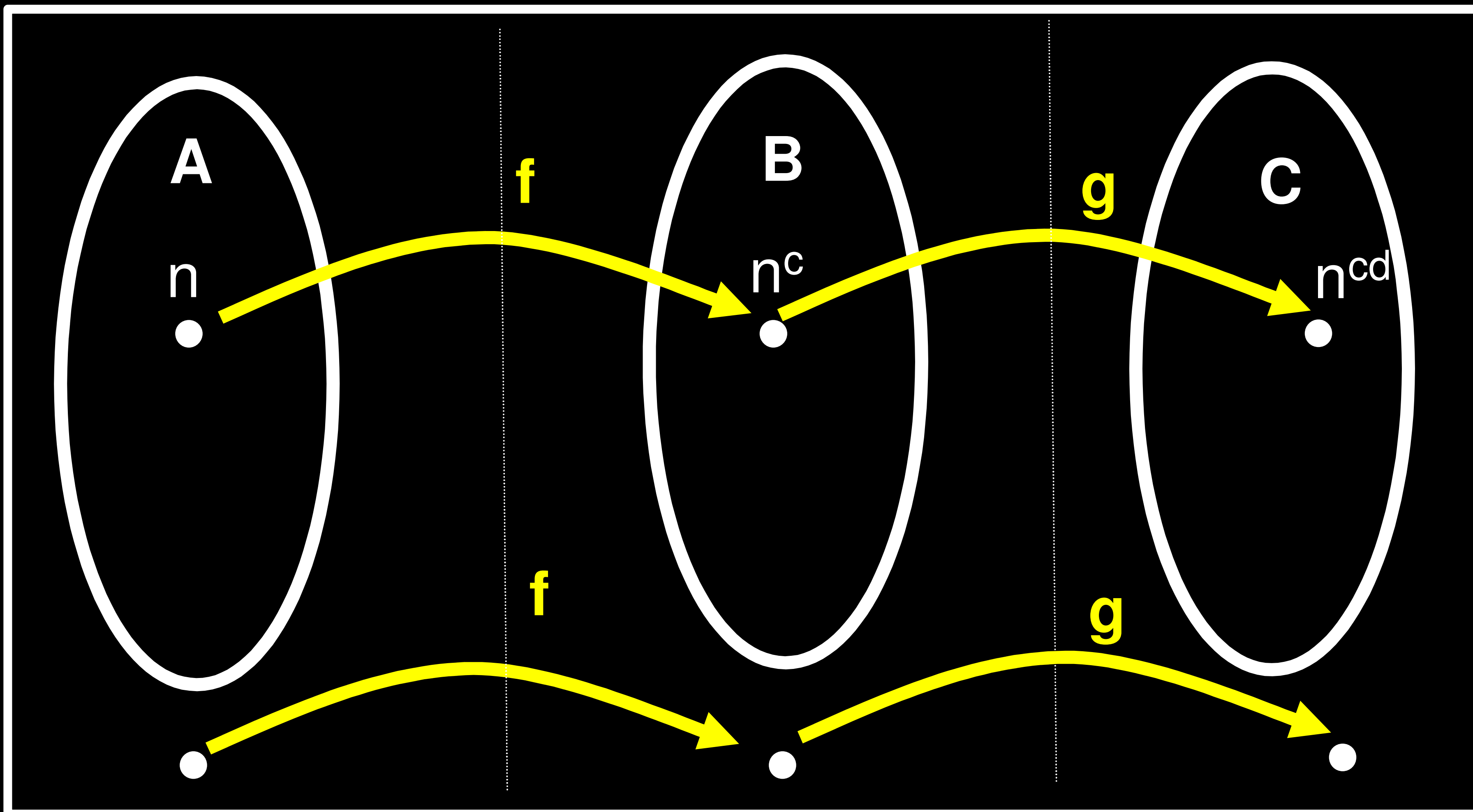
Language A is poly-time reducible to language B, written as $A \leq_P B$,
if there is a poly-time computable $f : \Sigma^* \rightarrow \Sigma^*$ so that:

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

f is a polynomial time reduction from A to B

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

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

We build a NTM M_A that decides A as follows:

M_A = On input w,

1. Compute f(w)
2. Run NTM M_B on f(w)

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

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

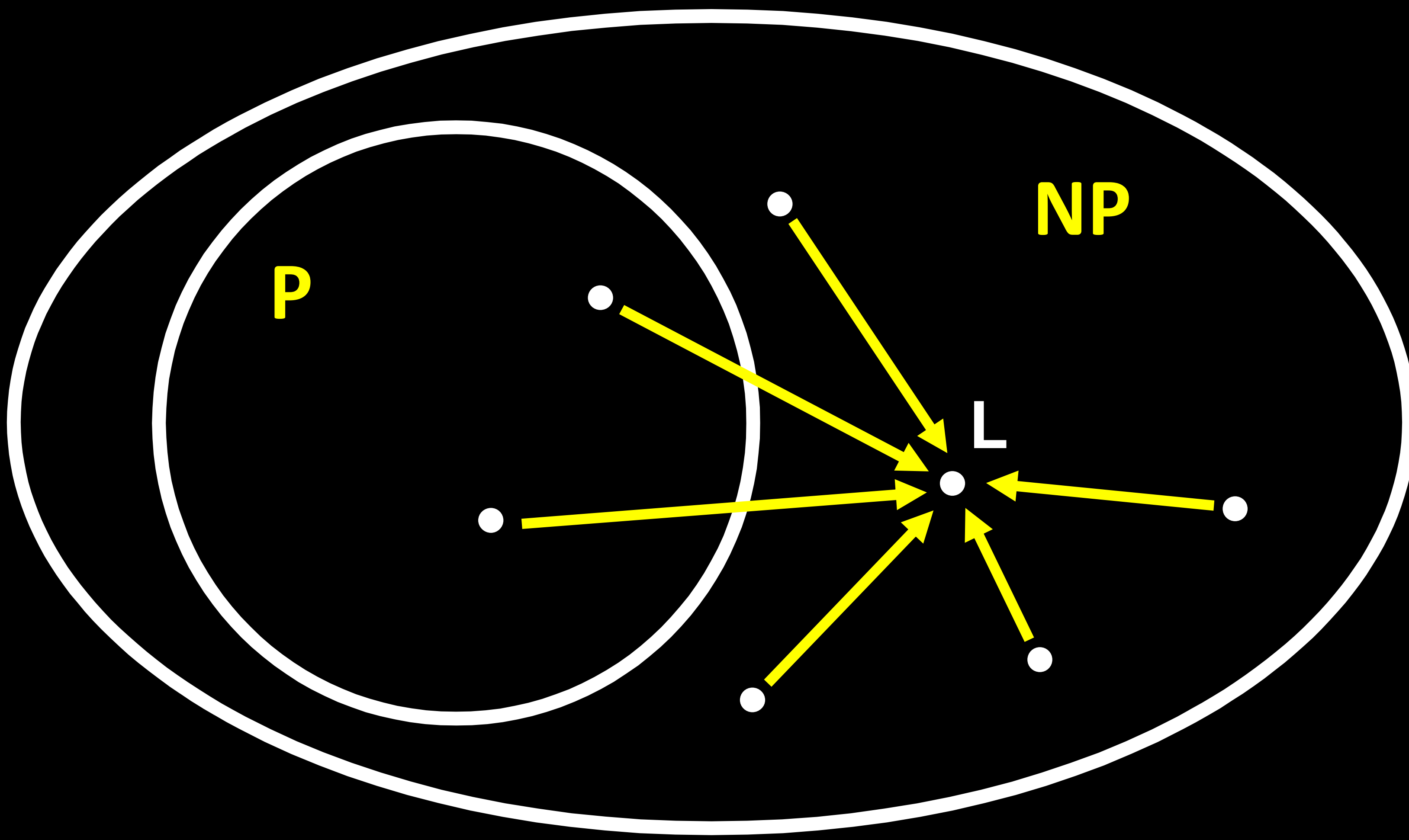
On your homework, you showed

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…



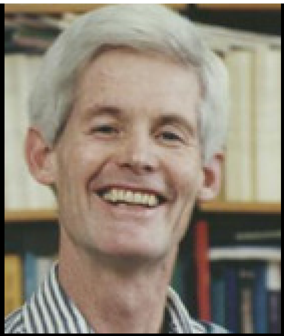
If $L \in P$, then $P = NP$!

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



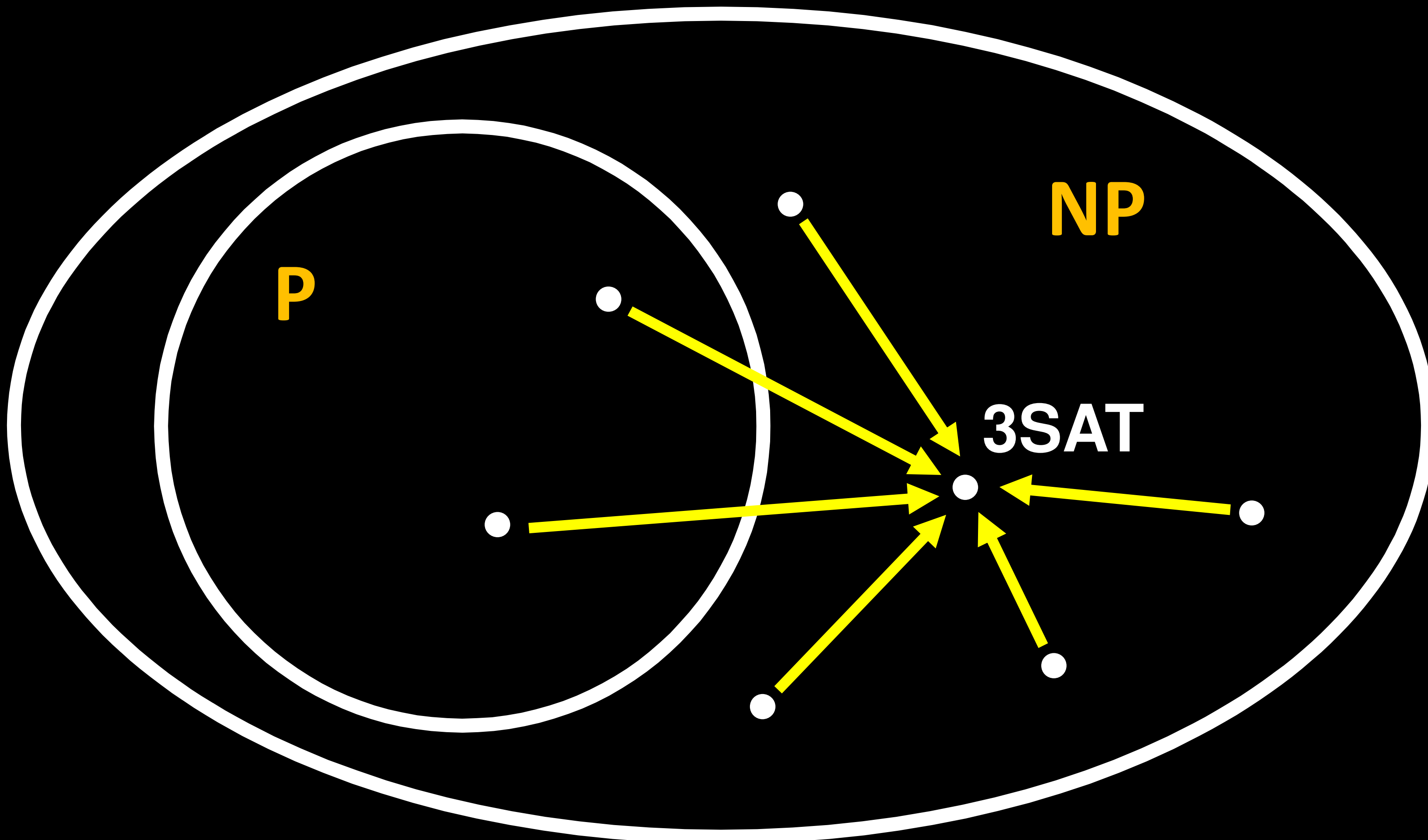
The Cook-Levin Theorem:
SAT and 3SAT are NP-complete

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

3SAT is NP-Complete



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

Proof Idea:

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

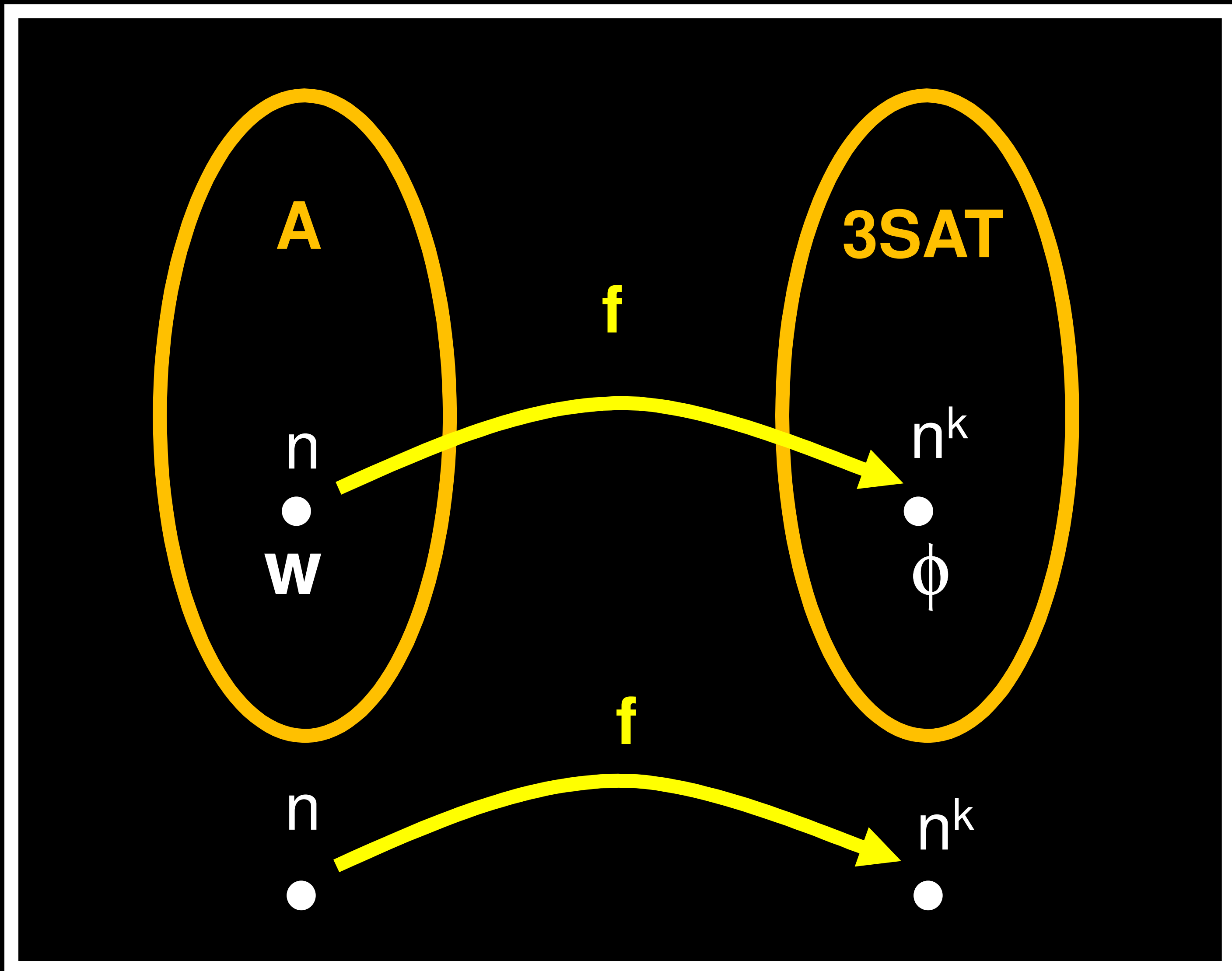
(2) Every language A in NP is polynomial time reducible to 3SAT (this is the challenge)

We give a poly-time reduction from A to SAT

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

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

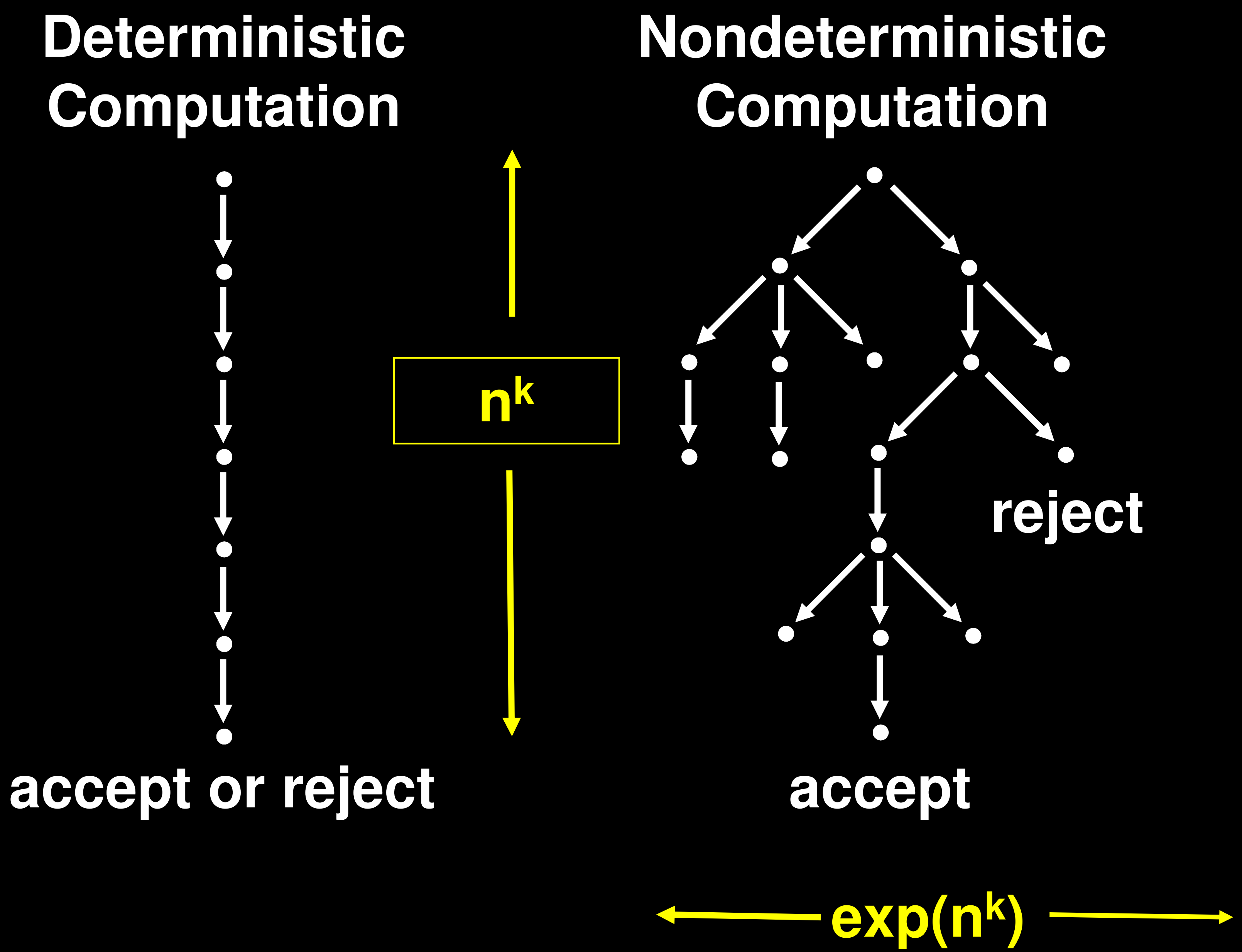
ϕ will simulate N on w



f turns any string w into a 3-cnf formula ϕ such that

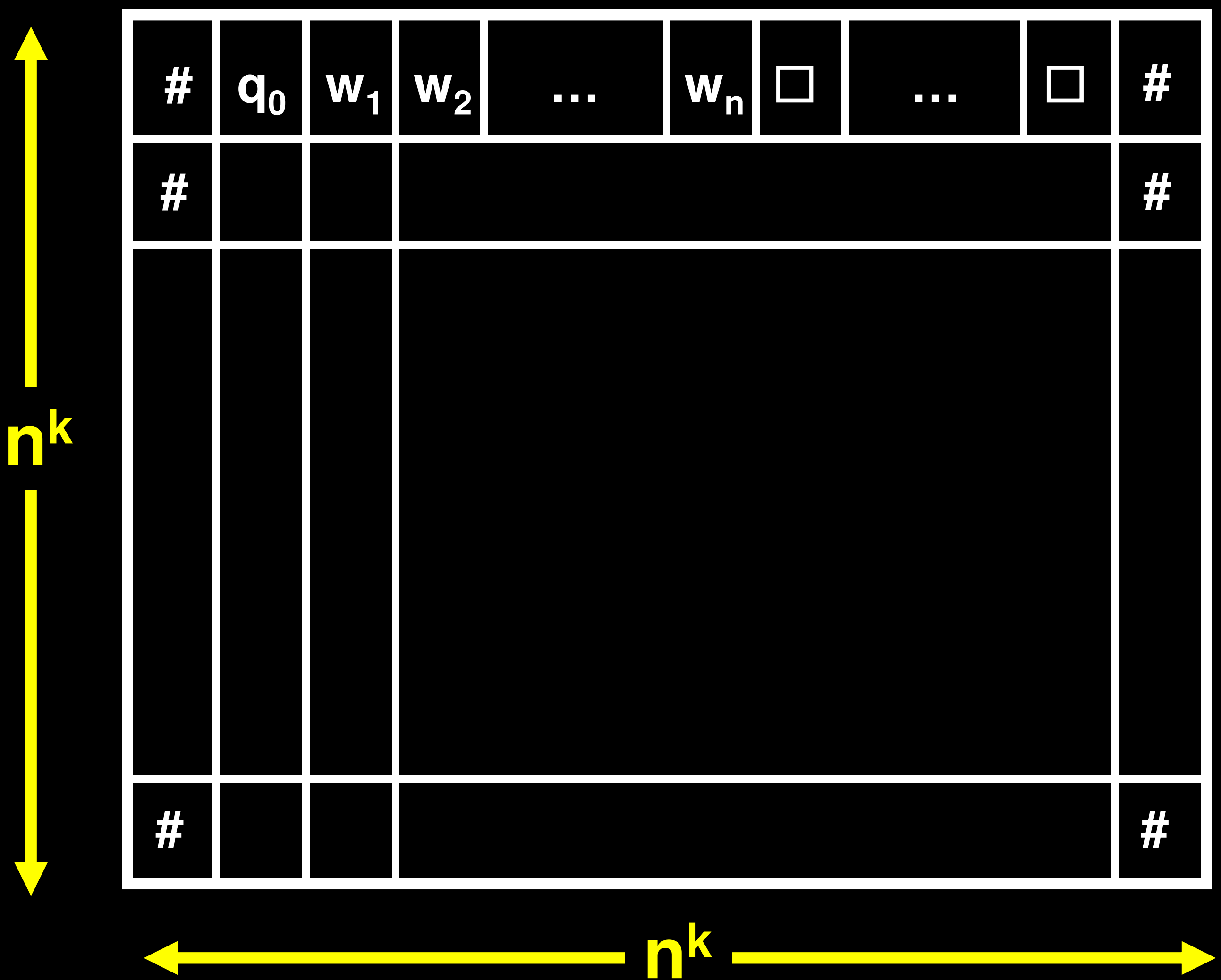
$w \in A \Leftrightarrow \phi$ is satisfiable

ϕ will simulate an NP machine N on w, where A = L(N)



Deterministic Computation
Nondeterministic Computation
n^k
reject
accept or reject
accept
$\exp(n^k)$

Let $L(N) \in NTIME(n^k)$. A **tableau for N on w** is an $n^k \times n^k$ table 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 $O(|w|^k)$ clauses, describing logical constraints that every 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 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 ($1 \leq i, j \leq n^k$) and for every $s \in C$ we have a Boolean variable $x_{i,j,s}$ in ϕ

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

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

We will have: 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:

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

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**