# CS301: Theory of Computation
# Section 7.4 - NP-Completeness

Daniel White

Gettysburg College
Computer Science

Spring 2024

In the 1970s, Stephen Cook and Leonid Levin discovered that some problems in NP had complexities related to the entire NP class. If any of these problems are in P, it would imply P = NP.

These problems are called **NP-complete**.

**Theory focus:** A researcher may attempt to demonstrate a single NP-complete problem is in P, thereby proving P = NP.

**Application focus:** This phenomenon provides a way to show various problems are "properly in NP" by creating a "polynomial reduction" from an NP-complete problem. Assuming P $\neq$ NP, this implies your problem is not computationally tractable.

homework: Know slideshow definitions, proofs, and exercises

## Definition (pg. 299)

A **Boolean formula** is an expression involving Boolean variables and operations. *e.g.* $\phi = (\bar{x} \wedge y) \vee (x \wedge \bar{z})$

## Definition (pg. 299)

A Boolean formula is **satisfiable** if some assignment of 0s and 1s to the variables makes the formula evaluate to 1.

The **satisfiability problem** is to test whether a Boolean formula is satisfiable. Consider

$$\text{SAT} = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula}\}.$$

**exercise:** Why does brute forcing not demonstrate SAT in P?
**exercise:** Argue SAT is in NP two different ways.

## Definition 7.28 (pg. 300)

A function $f : \Sigma^* \to \Sigma^*$ is a **polynomial time computable function** if some polynomial time Turing machine exists that halts with exactly $f(w)$ on its tape when started on input $w$.

## Definition 7.29 (pg. 300)

Language $A$ is **polynomial time reducible** to language $B$, expressed as $A \leq_P B$, if a polynomial time computable function exists where

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

The function $f$ is called a **polynomial time reduction** of $A$ to $B$.

## Theorem 7.31 (pg. 301)

If $A \leq_P B$ and $B \in P$, then $A \in P$.

*Proof.* Let $M$ be a polynomial time TM deciding $B$ and $f$ be a polynomial time reduction from $A$ to $B$. Consider the following polynomial time TM $N$ deciding $A$. On input $w$:

1. Compute $f(w)$.
2. Run $M$ on $f(w)$ and use $M$'s output.

Note that $M$ accepts $f(w)$ if and only if $w \in A$.

**exercise:** Explain why $N$ is polynomial time *relative to w's length.*

## Definition (pg. 302)

A **literal** is a Boolean variable or negated Boolean variable. A **clause** is several literals connected with $\vee$s, as in

$$(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4).$$

A Boolean formula is in **conjunctive normal form**, called a **cnf-formula**, if it comprises several clauses connected with $\wedge$s.

## Definition (pg. 302)

A cnf-formula where each clause has exactly three literals is called a **3cnf-formula**. Let 3SAT $= \{\langle\phi\rangle \mid \phi$ is satisfiable 3cnf-formula$\}$.

## Theorem 7.32 (pg. 302)

3SAT is polynomial time reducible to CLIQUE.

*Proof.* Let $\phi = \wedge_{i \leq k}(a_i \vee b_i \vee c_i)$ be a 3cnf-formula with $k$ clauses. We define a reduction $f$ to generate strings $\langle G, k \rangle$.

For each clause in $\phi$, create a triple $t_i$ of distinct nodes in a graph $G$. Label each node using the literals in that clause. Create edges that connect all but two types of pairs of nodes in G:

- no edges between nodes in same $t_i$
- no edges between two nodes with contradictory labels (*e.g.* $a_i = x$ and $b_j = \overline{x}$) ...

**exercise:** Finish this proof by showing that $\phi$ is satisfiable if and only if $G$ has a $k$-clique.

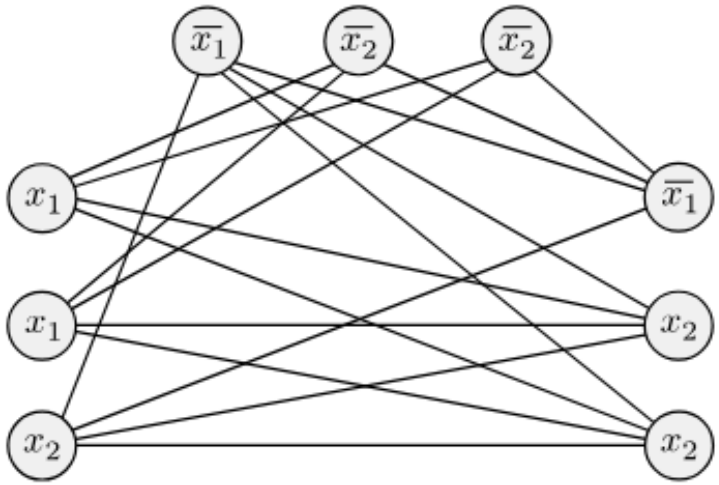


**FIGURE 7.33**
The graph that the reduction produces from
$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$

## Definition 7.34 (pg. 304)

A language $B$ is **NP-complete** if it satisfies two conditions:

1. $B$ is in NP, and
2. every $A$ in NP is polynomial time reducible to $B$

## Theorem 7.35 (pg. 304)

If $B$ is NP-complete and $B \in P$, then P $=$ NP.

## Theorem 7.36 (pg. 304)

If $B$ is NP-complete and $B \leq_{\mathrm{P}} C$ for $C$ in NP, then $C$ is NP-complete.

**exercise:** Write (short) proofs of the theorems above.

## Theorem 7.37 (Cook-Levin; pg. 304)

SAT is NP-complete.

*Proof idea.* We've already argued that SAT $\in$ NP. It remains to prove $A \leq_P$ SAT for all $A \in$ NP. We do so by translating the computational history of $A$'s polynomial NTM on input $w$ to a Boolean formula $\phi$, where $\phi$ is satisfiable if and only if $w \in A$.

*Proof.* Let $N$ be a NTM deciding $A$ in, at worst, $n^k - 3$ time. For each branch of $N$'s computation on a given input $w$, we may construct an $n^k \times n^k$ **tableau (2D table)** whose rows are the configurations along that branch. *See next slide ...*

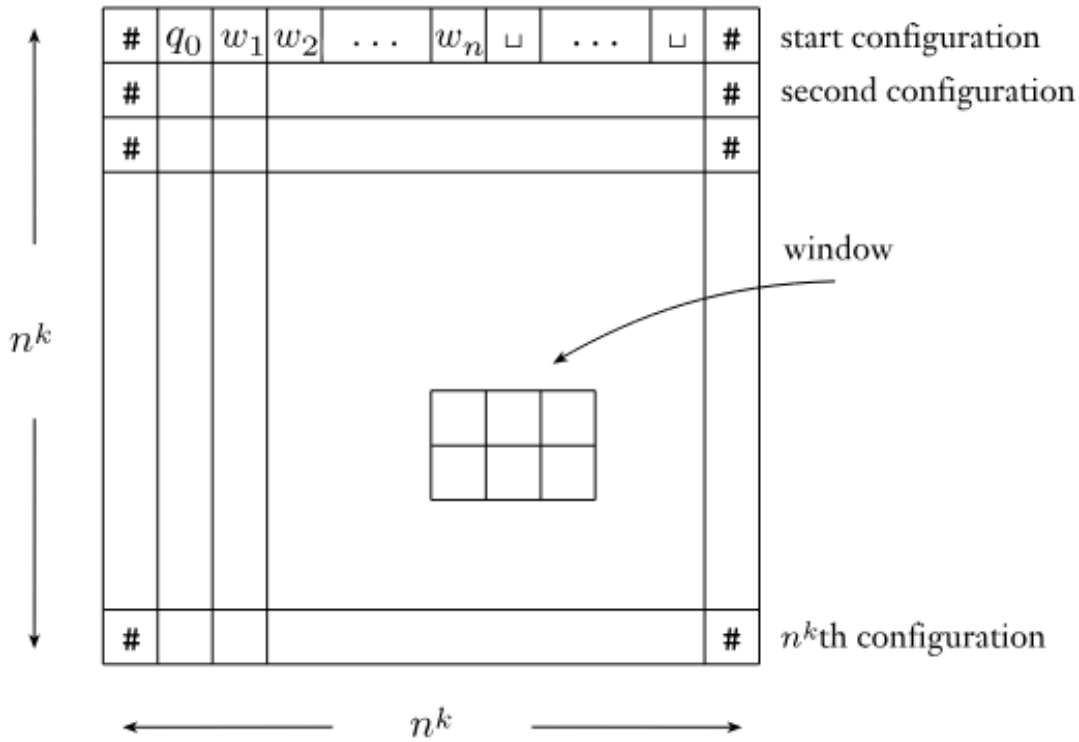


**FIGURE 7.38**
A tableau is an $n^k \times n^k$ table of configurations

Again, we aim to create a polynomial time mapping reduction from $A \in \text{NP}$ to SAT. Consider the Boolean variables

$$x_{i,j,s} := \text{cell}[i,j] \text{ in a tableau contains } s \in Q \cup \Gamma \cup \{\#\}.$$

We will generate a Boolean formula $\phi$ using the Boolean variables above such that $w \in A$ if and only if $\phi \in \text{SAT}$. This formula will be created in parts so that

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

where each constituent $\phi_?$ will be defined shortly.

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

$\phi_{\text{cell}}$ = detects if exactly one $s \in Q \cup \Gamma \cup \{\#\}$ occupies cell$[i, j]$,

$\phi_{\text{start}}$ = detects if the first row is the start configuration on $w$,

$\phi_{\text{move}}$ = detects if each $2 \times 3$ window in tableau is legal, and

$\phi_{\text{accept}}$ = detects if an accepting configuration is in tableau.

Note that $\phi$ is satisfied if and only if each $\phi_?$ above is satisfied, which occurs if and only if $w \in A$.

**exercise:** Write a formula for each $\phi_?$ in terms of the $x_{i,j,s}$.
**exercise:** Justify why this reduction is polynomial time.

**exercise:** Breathe. This course has concluded.