NP-Completeness

The P=NP Problem

Is $\mathbf{P} = \mathbf{NP}$?

To study this question we look at the *most difficult* problems in $\mathbf{NP}$, called $\mathbf{NP}$-complete problems.

SAT

A **Boolean formula** is a formula of propositional logic, constructed from variables and Boolean operations ($\wedge$, $\vee$, $\neg$)

A Boolean formula is **satisfiable** if there exists some assignment to the variables that makes the formula evaluate to 1

Example: $\phi = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$ is satisfiable. A satisfying assignment is $x = 1, y = 1, z = 0$

$\phi = x \wedge \overline{x} \wedge y$ is not satisfiable.

The Main Theorem

The **satisfiability problem** is the problem of deciding whether a given input Boolean formula is satisfiable

$SAT = \{\phi \mid \phi$ is a satisfiable Boolean formula $\}$

Theorem. **$SAT \in \mathbf{P}$ if and only if $\mathbf{P} = \mathbf{NP}$**

Polynomial Time Reductions

Definition. A function f is polynomial time computable if there exists a polynomial time TM that halts on each input x with only $f(x)$ on the tape.

Definition. A language A is polynomial time mapping reducible to a language B (write $A \leq_P B$) if A is mapping reducible to B via a polynomial time computable function.

3SAT

A **literal** is a variable or its negation

A **clause** is the disjunction of some literals, e.g., $x_1 \vee \overline{x_3} \vee x_{17}$

A Boolean formula is in **conjunctive normal form** if it is the conjunction ($\wedge$) of some clauses

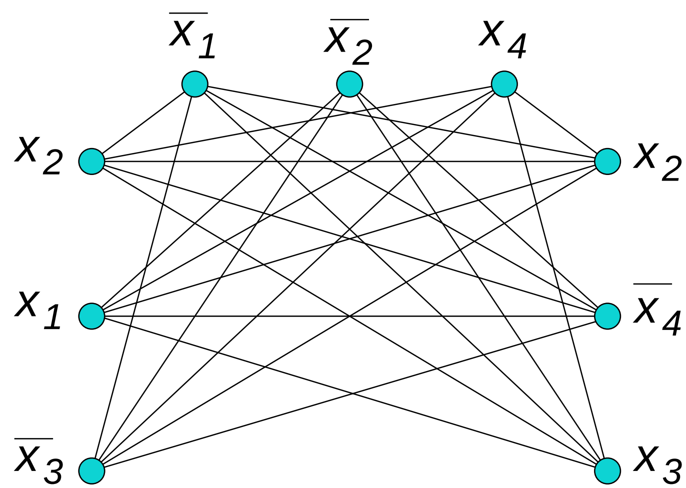
A **3CNF formula** is a formula in the CNF-form in which each clause consists of three literals

Theorem. ***3SAT* is polynomial time reducible to *CLIQUE*.**

Reducing 3SAT to CLIQUE

Given a formula ϕ of k three-literal clauses construct a graph $G = (V, E)$, where $V = \{\langle i, j\rangle \mid 1 \le i \le k, 1 \le j \le 3\}$ and $E = \{(\langle i, j\rangle, \langle i', j'\rangle) \mid (i \ne i')$ and (the jth literal in the ith clause) and (the j'th literal in the i'th clause) are **either identical to each other or use different variables**

The graph for the formula $\phi = (x_2 \vee x_1 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (x_2 \vee \overline{x_4} \vee x_3)$.



Reduction (cont'd)

Claim. **G has a k-clique if and only if ϕ is satisfiable.**

[$\Rightarrow$] Let S be a k-clique of G. Then S **has a node from each triple** so that **the selected nodes do not interfere with each other** as assignments.

[$\Leftarrow$] Let A be a satisfying assignment. Select from each triple **a literal that is satisfied by A** to construct a set S. $\|S\| = k$ and it is a clique.

The mapping is **polynomial time computable.** ∎

NP-Completeness

Definition. **A language A is NP-complete if A is in NP and every language in NP is polynomial time reducible to A.**

Theorem. **If a language A is NP-complete, then $A \in \mathrm{P}$ if and only if $\mathrm{P} = \mathrm{NP}$.**

We may use the following to prove something is $\mathbf{NP}$-complete.

Theorem. **A language A is NP-complete, $B \in \mathrm{NP}$, and $A \leq_P B$, then B is NP-complete.**

SAT is NP-Complete

Theorem. **SAT is $\mathbf{NP}$-complete.**

Proof $SAT \in \mathbf{NP}$. Consider a two-tape NTM that, on an input ϕ of n variables, **guesses and writes an assignment A** on Tape 2 using nondeterministic moves, accepts if $\phi(A) = 1$, and rejects $\phi(A) = 0$. Such a machine decides SAT and can be polynomial time.

The Converse

Suppose A is in $\mathbf{NP}$. We show $A \leq_P SAT$.

Note that there are some integer $k > 0$ and a one-tape NTM N such that

- $L(N) = A$, and
- for all inputs x, N on input x halts within $n^k + k$ steps.

By convention we assume that once N enters q_{accept}, N keeps moving its head to the right without changing the tape contents or state.

We will use $p(n)$ to denote $n^k + k + 2$.

Tableau

Let w be an input of length n. Define a **tableau** for N on w to be a $p(n) \times p(n)$ table whose rows are members of $\#\Gamma^*Q\Gamma^*\#$ with the following properties:

1. **the columns 1 and $p(n)$ are all $\#$,**

Tableau

Let w be an input of length n. Define a **tableau** for N on w to be a $p(n) \times p(n)$ table whose rows are members of $\#\Gamma^* Q \Gamma^* \#$ with the following properties:

1. **the columns 1 and $p(n)$ are all $\#$,**
2. **each row, excluding the first and the last symbols, is a configuration of N,**

Tableau

Let w be an input of length n. Define a **tableau** for N on w to be a $p(n) \times p(n)$ table whose rows are members of $\#\Gamma^* Q \Gamma^* \#$ with the following properties:

1. **the columns 1 and $p(n)$ are all $\#$,**
2. **each row, excluding the first and the last symbols, is a configuration of N,**
3. **the first row represents the initial configuration of N on w,**

Tableau

Let w be an input of length n. Define a **tableau** for N on w to be a $p(n) \times p(n)$ table whose rows are members of $\#\Gamma^* Q \Gamma^* \#$ with the following properties:

1. **the columns 1 and $p(n)$ are all $\#$,**
2. **each row, excluding the first and the last symbols, is a configuration of N,**
3. **the first row represents the initial configuration of N on w,** and
4. **for each i, $2 \leq i \leq p(n)$, the ith row results from the $(i-1)$st row applying one move of N**

Tableau

Let w be an input of length n. Define a **tableau** for N on w to be a $p(n) \times p(n)$ table whose rows are members of $\#\Gamma^* Q \Gamma^* \#$ with the following properties:

1. **the columns 1 and $p(n)$ are all $\#$,**
2. **each row, excluding the first and the last symbols, is a configuration of N with possible additional blanks at the end,**
3. **the first row represents the initial configuration of N on w,** and
4. **for each i, $2 \leq i \leq p(n)$, the ith row results from the $(i-1)$st row applying one move of N**

A tableau is **accepting** if

5. **the last row is an accepting configuration**

Encoding of Tableau

For all i, j, $1 \leq i, j \leq p(n)$, let $cell[i, j]$ denote the jth element in row i.

$cell[i, j]$ has more than 2 possibilities, and so, we introduce a boolean variable that represebnts each choice.

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

For each i, j, $1 \leq i, j \leq p(n)$, and $s \in C$, $x_{i,j,s}$ is the variable representing condition $cell[i, j] = s$.

We demand for all i, j, that $x_{i,j,s} =$ true for exactly one s.

General Technique

Let $F_1, \ldots, F_k$ be Boolean formulas. Then, "exactly one of $F_1, \ldots, F_k$ is true" can be expressed as another Boolean formula:

$$(F_1 \vee \cdots \vee F_k) \wedge \neg((F_1 \wedge F_2) \vee (F_1 \wedge F_3) \vee \cdots \vee (F_{k-1} \wedge F_k))$$

So, we have a formula for conditions like $cell[i,j] = a$ and $cell[i,j] \in S$ where S is a finite set of symbols.

Plan

We will encode Conditions 1 through 5 into Boolean formulas ϕ_1 through ϕ_5 and then construct $\phi_x = \phi_1 \wedge \cdots \wedge \phi_5$. We will map x to ϕ_x.

Condition 1: “#’s”

$$\bigwedge_{1 \le i \le p(n)} \left(cell[i,1] = \# \wedge cell[i,p(n)] = \# \right).$$

Condition 5: "Accepting"

$$\bigvee_{1 \le j \le p(n)} cell[p(n), j] = q_{\mathrm{accept}}$$

Condition 3: "Initial"

$$
\begin{aligned}
& cell[1,2] = q_0 \\
\wedge & \left(\bigwedge_{1 \le j \le n} cell[1, 2+j] = w_j \right) \\
\wedge & \left(\bigwedge_{n+3 \le j \le p(n)-1} cell[1, j] = \sqcup \right)
\end{aligned}
$$

Condition 2: "Configuration"

$$\bigwedge_{1 \le i \le p(n)} \left(\bigwedge_{j, 2 \le j \le p(n)-1} (cell[i,j] \in Q \cup \Gamma) \wedge B_i \right)$$

where $B_i =$ "there is exactly one $j, 2 \le j \le p(n) - 1$ such that $cell[i,j] \in Q$"

Condition 4: "Step"

This is

$$\bigwedge_{2 \leq i \leq p(n)} \bigwedge_{2 \leq j \leq p(n)-1} D_{ij}$$

Here D_{ij} is the formula that states: In the 2×3 block of the tableau

$cell[i-1, j-1]$	$cell[i-1, j]$	$cell[i-1, j+1]$
$cell[i, j-1]$	$cell[i, j]$	$cell[i, j+1]$

(α_{ij}) if $cell[i-1, j]$ is in Q then the six cells encode the outcome of a permissible action of N; and

(β_{ij}) "if $cell[i-1, j], cell[i-1, j-1], cell[i-1, j+1] \notin Q$ then $cell[i-1, j-1] = cell[i, j-1]$, $cell[i-1, j] = cell[i, j]$, and $cell[i-1, j+1] = cell[i, j+1]$."

The entire formula has length bounded a fixed polynomial in n. ∎

3SAT is NP-Complete

The formula in the tableau method can be **converted to an equivalent 3CNF formula**

The formula constructed in the previous proof is trivially in CNF except for the transition part, which is expressed as the conjunction of D_{ij}. Here D_{ij} is $\alpha_{ij} \wedge \beta_{ij}$ and checks that the 2-by-3 block located at (i, j) is valid.

D_{ij} can be expressed either as

- the six cells are in one of valid combinations
- the six cells are not in any of invalid combinations.

There are only a constant number of invalid combinations, so we will use the latter.

Conversion

Suppose

$$(cell[i-1,j-1]=a, cell[i-1,j]=b, cell[i-1,j+1]=c,$$
$$cell[i,j-1]=d, cell[i,j]=e, cell[i,j+1]=f)$$

is an in valid form. Then

$$(\overline{x_{i-1,j-1,a}} \vee \overline{x_{i-1,j,b}} \vee \overline{x_{i-1,j+1,c}}$$
$$\vee \overline{x_{i,j-1,d}} \vee \overline{x_{i,j,e}} \vee \overline{x_{i,j+1,f}})$$

expresses that these cells are not in that combination.

By taking the conjunction for all invalid combinations, we obtain a CNF formula for D_{ij}.

Converting CNF to 3CNF

Conversion rules:

1. $(x \wedge y) \wedge z$ is equivalent to $x \wedge y \wedge z$
2. $(x \vee y) \vee z$ is equivalent to $x \vee y \vee z$
3. $(x \vee y \vee z \vee u)$ is equivalent to $(x \vee y \vee w) \wedge (w \equiv (z \vee u))$. The second term is equivalent to $(\overline{w} \vee z \vee u) \wedge (\overline{z} \vee w) \wedge (\overline{u} \vee w)$

Repeat literals in clauses with < 3 literals to make the number of literals equal to three.