**NP-Completeness (Contd.)**

**Cook's Thm** CSAT is NP-hard.

Recall this provides us the starting point for showing a large number of new problems are NP-hard (via poly-time reductions).

Convenient to use 3-SAT as the canonical starting point of reductions, so first we apply Cook's Thm to showing that 3-SAT is NP-hard.

**Definitions Reviewed**

**CSAT** SAT for CNF (Conjunctive Normal Form) formulas.

**Notation** + is $\lor$ (Or),

$x$ is $\land$ (And).

**Literal** $x_i$ or $\overline{x}_i$ (Variable or its negation).

**Clause** "Or" of literals

$$C_i = \sum_{j} L_{ij}, \text{ where } L_{ij} \text{ is } j^{th} \text{ literal in clause } C_i$$

**CNF Formula** "And" of clauses

$$F(x_1, \ldots, x_n) = \prod_{i=1}^{m} C_i$$

**Example**

$$F = (x_2 + \overline{x}_3)(x_1 + x_4 + \overline{x}_5)$$
CSAT PROBLEM  GIVEN CNF FORMULA $F(x_1, \ldots, x_n)$, IS IT SATISFIABLE?

CLEARLY SAT $\in$ NP $\Rightarrow$ CSAT $\in$ NP

COOK'S THEOREM CSAT IS NP-HARD

COROLLARY SAT IS NP-HARD

(SINCE, TRIVIALLY, CSAT $\leq_{\text{poly}}$ SAT)

$R$-CNF FORMULA CNF FORMULA $F(x_1, \ldots, x_n)$ WHERE EACH CLAUSE HAS EXACTLY $R$ LITERALS.

$R$-SAT PROBLEM GIVEN $R$-CNF FORMULA $F(x_1, \ldots, x_n)$, IS IT SATISFIABLE?

FACT 1-SAT $\in$ P

EXAMPLE $(x_1), (\overline{x_2}), (x_3), \ldots$

FACT 2-SAT $\in$ P

EXAMPLE $(x_1 + \overline{x_2}), (x_2 + \overline{x_3}), (x_3 + \overline{x_1})$

3-SAT

THEOREM 3-SAT IS NP-COMPLETE

PROOF (a) 3-SAT $\in$ NP TRIVIALLY, SINCE SAT $\leq_{\text{poly}}$ 3-SAT

(b) 3-SAT IS NP-HARD

IDEA: SHOW CSAT $\leq_{\text{poly}}$ 3-SAT
**CSAT \leq_{poly} 3-SAT**

**REDUCTION**

**INPUT** CNF FORMULA \( F(x_1, x_2, \ldots, x_k) \)

**OUTPUT** 3-CNF FORMULA

\[
G(x_1, \ldots, x_k, y_1, \ldots, y_\ell)
\]

WHERE \( y_1, \ldots, y_\ell \) ARE NEW VARIABLES

**IDEA**

**SUPPOSE** \( F(x_1, \ldots, x_k) = C_1 \land C_2 \land \ldots \land C_m \)

IN \( G \) WILL ADD NEW VARIABLES \( y_1, \ldots, y_\ell \) AND REPLACE EACH CLAUSE \( C_i \) BY A NEW COLLECTION OF CLAUSES, EACH OF WHICH HAS EXACTLY 3 LITERALS.

ENSURING THAT

\( \exists \ T.A. \ T_F \) SATISFYING \( F(x_1, \ldots, x_k) \)

\( \iff \exists \ T.A. \ T_G \) SATISFYING \( G(x_1, \ldots, x_k, y_1, \ldots, y_\ell) \)

WHERE

\( T_F \): T.A. GIVING VALUES TO \( x_1, \ldots, x_k \)

\( T_G \): T.A. GIVING VALUES TO \( x_1, \ldots, x_k, y_1, \ldots, y_\ell \)

**IN PARTICULAR**

**SUPPOSE** \( T_F \) SATISFIES \( F \) WILL SHOW HOW TO ASSIGN VALUES TO \( y_1, \ldots, y_\ell \) SO TOGETHER THIS SATISFIES \( G \).

**SUPPOSE** \( T_G \) SATISFIES \( G \) WILL SHOW THAT \( T_G \) WHEN RESTRICTED TO \( x_1, \ldots, x_k \) SATISFIES \( F \).
CONSTRUCTING 3-CNF $G$?

**Note:** We will show how to replace each clause $c_i$ from $F$ by a collection of clauses in $G$.

It will be clear that a halting T.M. can perform this replacement in poly-time (actually, linear).

**Consider** clause $c_i = z_1 + z_2 + \ldots + z_k$ in $F$ where literals $z_j$ are from $x_1, \ldots, x_r$ (or their negations).

**4 cases:** $k = 1, 2, 3, t > 3$ considered separately.

**Case I** ($k = 3$)

Trivial: $c_i$ is already in 3-CNF form, so replace by itself, i.e. no change.

**Case II** ($k = 2$)

Here $c_i = z_1 + z_2$

Add new variable $y$.

Replace $c_i$ by 2 new clauses:

1. $z_1 + z_2 + y$
2. $z_1 + z_2 + \overline{y}$

Clearly $T_F$ satisfies these only by choosing $x_1, \ldots$ such that $z_1 + z_2$ is satisfied.

Thus $\exists T_F$ satisfying $c_i \iff \exists T_G$ which satisfies both replacing clauses.
**Case III** \( [k=1] \)

Here \( c_k = z_1 \)

Add 2 new variables \( y_1, y_2 \)

Replace \( c_k \) by 4 new clauses

\[
\begin{align*}
& z_1 + y_1 + y_2 \\
& z_1 + y_1 + \overline{y_2} \\
& z_1 + \overline{y_1} + y_2 \\
& z_1 + \overline{y_1} + \overline{y_2}
\end{align*}
\]

Clearly \( T_0 \) satisfies all of these only by choosing \( x_1, \ldots, x_r \) such that \( z_1 \) is true

Thus \( \exists T \) satisfying \( c_k \iff \forall T_0 \) which satisfies all 4 replacing clauses

**Remark** We are considering all clauses in an independent manner — but this is fine since all clauses need to be satisfied

**Case IV** \( [k>3] \)

Here \( c_k = z_1 + z_2 + \ldots + z_k \) with \( k > 3 \)

Note this is the hard case since we need to simulate \( c_k \)'s effect by "smaller" clauses

Idea: Add \( k-3 \) new variables \( y_1, y_2, \ldots, y_{k-3} \)
REPLACE $C_i$ BY $t-2$ NEW CLAUSES

- $z_1 + z_2 + y_1$
- $z_3 + \overline{y_1} + y_2$
- $z_4 + \overline{y_2} + y_3$
- ...
- $z_{t-2} + \overline{y}_{t-4} + y_{t-3}$
- $z_{t-1} + z_t + \overline{y}_{t-3}$

EXAMPLE

$F(x_1, x_2, x_3, x_4, x_5) = (x_1 + \overline{x_2} + \overline{x_3} + x_4 + y_5) (\overline{x_1} + x_6)$

THEN

$G(x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3) =$

$(x_1 + \overline{x_2} + y_1) (\overline{x_3} + \overline{x_1} + y_2) (x_4 + x_5 + \overline{y_3})$

$(\overline{x_1} + x_4 + y_3) (\overline{x_1} + x_4 + \overline{y_3})$

CLAIM CASE IV WORKS, i.e. GIVEN TF FOR F WE CAN INFER TG FOR G, AND VICE VERSA

PROOF $(TF \Rightarrow TG)$

SUPPOSE $TF$ SATISFIES $F(x_1, \ldots, x_n)$

$\Rightarrow TF$ MAKES $C_i$ EVALUATE TO TRUE

$\Rightarrow TF$ ASSIGNS VALUE "TRUE" TO SOME LITERAL $z_i$ IN $C_i$
Obtaining $T_G$

$X_1, \ldots, X_k$ get same values as in $T_F$

$Y$ values we only need to focus on $Y$-variables appearing in clauses replacing $C_i$:

- These are assigned as follows:

\[
\begin{align*}
Z_1 + Z_2 + Y_1 & \quad \text{SET } Y_1, Y_2, \ldots, Y_{j-2} \text{ TO TRUE}
\end{align*}
\]

\[
\begin{align*}
Z_3 + \overline{Y}_1 + Y_2 & \quad \text{thereby satisfying these clauses}
\end{align*}
\]

\[
\begin{align*}
\vdots
\end{align*}
\]

\[
\begin{align*}
Z_{j-1} + \overline{Y}_{j-3} + Y_{j-2} & \quad \text{set } Z_j \text{ to true}
\end{align*}
\]

\[
\begin{align*}
Z_j + \overline{Y}_{j-2} + Y_{j-1} & \quad \text{satisfied since } T_F
\end{align*}
\]

\[
\begin{align*}
Z_{j+1} + \overline{Y}_{j-1} + Y_j & \quad \text{set } Y_{j-1}, Y_j, \ldots, Y_{k-3} \text{ to false}
\end{align*}
\]

\[
\begin{align*}
Z_{j+2} + \overline{Y}_j + Y_{j+1} & \quad \text{thereby satisfying these clauses}
\end{align*}
\]

\[
\begin{align*}
\vdots
\end{align*}
\]

\[
\begin{align*}
Z_{t-1} + Z_t + \overline{Y}_{t-3} & \quad \text{set } Y_{j-1}, Y_j, \ldots, Y_{k-3} \text{ to false}
\end{align*}
\]

Remark: In cases I & II the new variables were assigned arbitrary values by $T_G$, but here we need specific values.
PROOF \((T_G \Rightarrow T_F)\)

**Suppose** all \(t-2\) clauses are satisfied by \(T_G\)

**Then** we claim that \(T_G\) must assign \(x\)-values in such a way that at least one \(z_j\) in \(C_1\) is set to true.

**Why?**

**Suppose** all \(z_j\)'s are false

**Then**

- \(z_1 + z_2 + \bar{y}_1\) needs \(y_1 = \text{true}\)
- \(z_3 + \bar{y}_1 + y_2\) needs \(y_2 = \text{true}\)
- \(z_4 + \bar{y}_2 + y_3\) needs \(y_3 = \text{true}\)

... and so on.

- \(z_{t-2} + \bar{y}_{t-4} + y_{t-3}\) needs \(y_{t-3} = \text{true}\)
- \(z_{t-1} + z_t + \bar{y}_{t-3}\) problem not satisfied!
**CLIQUE**

Instance graph \( G(V, E) \), integer \( t \)

Problem: Does \( G \) have a **clique** of size \( \geq t \)?

Recall: "**clique**" is a set of vertices \( S \subseteq V \) such that for any \( u, v \in S \), the edge \( (u, v) \) belongs to \( G \).

(That is, a complete subgraph.)

<table>
<thead>
<tr>
<th>Clique Size</th>
<th>Clique</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="Clique 1" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image2" alt="Clique 2" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image3" alt="Clique 3" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="image4" alt="Clique 4" /></td>
</tr>
</tbody>
</table>

**Example**

\( G(V, E) \)

3-clique: \( S = \{3, 4, 6\} \) or \( \{1, 2, 3\} \) or \( \{1, 3, 5\} \) or \( \{1, 2, 5\} \) or \( \{2, 3, 5\} \)

4-clique: \( S = \{1, 2, 3, 5\} \)
Theorem: Clique is NP-complete

Proof:
(a) Clique ∈ NP

\[ \text{NTM } N \xleftarrow{\text{Guess}} S \subseteq V, \text{ with } |S| = t \]

\[ \text{Verify all } \left( \begin{array}{c} t \\ 2 \end{array} \right) \text{ edges of } S \]

Easily \( O(n^3) \) time or better

(b) Clique is NP-hard

Idea: Show \( I.S. \leq_{\text{Poly}} \text{ clique} \)

Observe we can define \( G \) as the complement of graph \( G \), as follows:

- For every pair of vertices \( u, v \in V \), the edge \( (u, v) \in \overline{G} \) if and only if the edge \( (u, v) \notin G \).

Example (continued)

\[ G \]

\[ \overline{G} \]

Observe cliques in \( G \) become independent sets in \( \overline{G} \), and vice versa.

Reduction:

Input instance of I.S., which is a graph \( G(V, E) \) and integer \( t \)

Output instance of Clique, which is a graph \( G'(V', E') \) and integer \( k \)
REDN. A SETS \( G' = \overline{G} \) AND \( t = r \)

CLEARLY

(a) \( \emptyset \) COMPUTABLE BY A DTM IN POLY-TIME

(ACTUALLY, LINEAR)

(b) \( G \) HAS I.S. OF SIZE \( r \) \( \implies \) \( \overline{G} \) HAS CLIQUE OF SIZE \( r \)

DONE

---

VERTEX COVER

INSTANCE

GRAPH \( G(V, E) \), INTEGER \( t \)

PROBLEM

DOES \( G \) HAVE A V.C. OF SIZE \( \leq t \)?

RECALL

VERTEX COVER IS A SUBSET \( S \subseteq V \) SUCH THAT EACH EDGE IN \( G \) HAS AT LEAST ONE END-POINT IN \( S \).

EXAMPLE

\[
\begin{array}{c}
1 \\
5 \\
2 \\
3 \\
4 \\
6
\end{array}
\]

POSSIBLE V.C. \( \{1, 3, 5, 6\} \), \( \{2, 4, 5, 6\} \)

CLAIM

FOR ANY V.C. \( S \subseteq V \), THE COMPLEMENT SET \( \overline{S} = V - S \) IS AN INDEPENDENT SET.
**Proof**

Consider any \( u, v \in S = V - S \)

**Suppose** the edge \((u, v)\) is present in \(G\)

**Then** \(S\) cannot be a vertex cover,

since neither of \(u, v\) is in \(S\).

**Thus** \(S\) is an independent set.

---

**Theorem** V.C. is \(\text{NP}-\text{complete}\)

**Proof**

(a) V.C. \(\in\text{NP}\)

**Guess** \(S \subseteq V\), with \(|S| = \ell\)

**NTM** \(N\) \(<\) **Verify** all edges in \(G\) have one end-point in \(S\)

**Easily** \(O(n^3)\) time, or better.

(b) V.C. is \(\text{NP-hard}\)

**Idea** show \(I.S. \leq_{\text{poly}} \text{V.C.}\)

**Reduction** \(\Theta\) — input instance of \(I.S.\), a graph \(G(V, E)\) and integer \(k\)

**Output** instance of V.C., a graph \(G'(V', E')\) and integer \(\ell\).

**Computing** \(\Theta\) — set \(G' = G\) and \(\ell = n - k\)

(where \(n = |V|\))

**Clearly** computable in poly-time (linear)

**Fact** \(G\) has I.S. of size \(k \iff G\) has a V.C. of size \(n-k\).