**NP Completeness (cont’d)**

**Reductions:**
- Let $L_1$ and $L_2$ be languages.
- We say that $L_1$ is \textit{poly-time reducible to} $L_2$, (notation: $L_1 \leq_P L_2$) if there exists a poly-time computable function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ such that:

\[
\forall x \in \{0, 1\}^* : x \in L_1 \iff f(x) \in L_2
\]

**Definition of \textbf{NP}:**
- \textbf{NP} is the class of languages $L$ such that for some $L' \in \textbf{P}$ and some constants $a, b, c$:

\[
\forall x \in \{0, 1\}^* : \quad x \in L \iff \exists w \in \{0, 1\}^{a|x|b+c} : \langle x, w \rangle \in L'
\]
Definition of NP-completeness:

- A language $L$ is called $NP$-complete if
  1. $L \in \textbf{NP}$, and
  2. $L$ is $NP$-hard: for all $L' \in \textbf{NP}$: $L' \leq_P L$

Formalizing computation:

- Define an idealized model of computation
- RAM: Random Access Machine
- Reads bits from an input tape
- Writes bits to an output tape
- Random access memory
- Simple instruction set
Random Access Machine (RAM)

Random Access Memory (RAM)
Instruction Set:

- **add 17, 18, 20 # m[20] ← m[17] + m[18]**
- **sub 17, 18, 20 # m[20] ← m[17] − m[18]**
- **mul 17, 18, 20 # m[20] ← m[17] · m[18]**
- **div 17, 18, 20 # m[20] ← ⌊m[17]/m[18]⌋**
- **ldc 17, 20 # m[20] ← 17**
- **ldd 17, 20 # m[20] ← m[17]**
- **ldi 17, 20 # m[20] ← m[m[17]]**
- **sti 17, 20 # m[m[20]] ← m[17]**
- **b 100 # branch to 100**
- **bpos 17, 100 # branch to 100 if m[17] > 0**
- **bz 17, 100 # branch to 100 if m[17] = 0**
- **halt**
- **read 20 # m[20] ← read bit**
- **write 17 # write m[17]**
Polynomial time:

- $n = \text{input length}$
- Requirement: Number of instructions executed $\leq an^b + c$ for constants $a, b, c$
- Requirement: Number in each memory cell $\leq a'n^{b'} + c'$ in absolute value for constants $a', b', c'$
- Implication: highest memory cell addressed is $\leq a'n^{b'} + c''$ for constant $c''$
Circuit Satisfiability (CSAT) 
a first NP-complete problem

Instance:

- A Boolean circuit $C$:
  - inputs $x_1, \ldots, x_m$
  - constant gates (0, 1)
  - AND, OR, NOT gates
  - AND, OR take 2 inputs
  - unrestricted “fan out”
  - A single bit output

Question:

- Is there an assignment to the inputs $x_1, \ldots, x_m$ such that $C(x_1, \ldots, x_m) = 1$?
Linearized representation:

\[
\begin{align*}
    x_4 & \leftarrow x_1 \land x_2 \\
    x_5 & \leftarrow \overline{x_1} \\
    x_6 & \leftarrow x_3 \lor x_4 \\
    x_7 & \leftarrow x_4 \lor x_5 \\
    x_8 & \leftarrow x_6 \land x_7
\end{align*}
\]
Proof that CSAT is NP-complete

• Clearly $CSAT \in \text{NP}$: a witness is just a satisfying assignment

• Need to show that $CSAT$ is $NP$-hard, i.e., $L \leq_p CSAT$ for all $L \in \text{NP}$

• Let $L \in \text{NP}$

• We know there is a language $L' \in \text{P}$ and constants $a, b, c$ such that $\forall x \in \{0, 1\}^* :$
  
  $x \in L \iff \exists w \in \{0, 1\}^{a|x|^b+c} : \langle x, w \rangle \in L'$

• Let $M'$ be the polynomial time RAM that recognizes $L'$
Proof (cont’d):
• The current configuration of $M'$ is 
  $\alpha = (m, p, r, y, z)$, where
  ○ $m$: contents of all memory cells
  ○ $p$: program counter
  ○ $r$: position of input “read head”
  ○ $y$: contents of input tape
  ○ $z$: contents of output tape
• There is a function $f_{\text{next}}$ that maps a
  configuration $\alpha$ to the successor configuration
  $f_{\text{next}}(\alpha)$
• Configurations can be encoded as
  polynomial-sized bit strings
• The function $f_{\text{next}}$ can be realized by a
  polynomial-sized circuit $C_{\text{next}}$
input: \( w \)

\[
\langle x, \cdot \rangle
\]

``pairing circuit''

x is ``hardwired''

\[
\alpha_0
\]

\[
\alpha_1
\]

\[
\alpha_t
\]

\[
\text{output}
\]
Satisfiability (SAT)

Instance:
- A Boolean formula $\phi$:
  - variables $x_1, \ldots, x_m$
  - constants $0, 1$
  - operators $\lor, \land, \bar{\cdot}$
  - Parentheses

Question:
- Is there an assignment to the variables $x_1, \ldots, x_m$ such that $\phi(x_1, \ldots, x_m) = 1$?

Formulas are essentially circuits with fan-out restricted to 1
A simple reduction: $CSAT \leq_P SAT$

- Let "\( \phi_1 \iff \phi_2 \)" be shorthand for "\((\phi_1 \land \phi_2) \lor (\bar{\phi}_1 \land \bar{\phi}_2)\)"

Circuit $C$: 

<table>
<thead>
<tr>
<th>Circuit C:</th>
<th>Formula $\phi$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_4 \leftarrow x_1 \land x_2 )</td>
<td>( (x_4 \iff (x_1 \land x_2)) \land )</td>
</tr>
<tr>
<td>( x_5 \leftarrow \bar{x}_1 )</td>
<td>( (x_5 \iff (\bar{x}_1)) \land )</td>
</tr>
<tr>
<td>( x_6 \leftarrow x_3 \lor x_4 )</td>
<td>( (x_6 \iff (x_3 \lor x_4)) \land )</td>
</tr>
<tr>
<td>( x_7 \leftarrow x_4 \lor x_5 )</td>
<td>( (x_7 \iff (x_4 \lor x_5)) \land )</td>
</tr>
<tr>
<td>( x_8 \leftarrow x_6 \land x_7 )</td>
<td>( (x_8 \iff (x_6 \land x_7)) \land )</td>
</tr>
</tbody>
</table>

- It is clear that $C$ is satisfiable $\iff \phi$ is satisfiable
- Note: $C$ and $\phi$ are not logically equivalent
3SAT: a special case of SAT

Conjunctive Normal Form:

- a conjunction (\( \land \)) of clauses
- each clause is a disjunction (\( \lor \)) of literals
- each literal is a variable \( x \) or its complement \( \overline{x} \)

Examples:

\[
x \land y, \quad \overline{x} \land (y \lor z), \quad (x \lor y \lor \overline{z}) \land (w \lor \overline{x} \lor z)
\]

A special form: 3-CNF

- Each clause consists of 3 distinct literals
- \( 3SAT := \left\{ \langle \phi \rangle : \phi \text{ is a satisfiable 3-CNF formula} \right\} \)
Fact: every formula $\psi$ in 1–3 variables can be rewritten as a 3-CNF formula (with at most 8 clauses)

- Add extra variables to make $\#$ of variables $= 3$
- Write down truth table for $\overline{\psi}$
- Read off disjunctive normal form formula from truth table
- Negate this formula, using DeMorgan’s law to get 3-CNF
Proof that 3SAT is NP-hard

- Reduction: $\text{CSAT} \leq_P \text{3SAT}$
- Let $N(\psi)$ be a 3-CNF formula representing $\psi$

Circuit $C$:

- $x_4 \leftarrow x_1 \land x_2$
- $x_5 \leftarrow \overline{x_1}$
- $x_6 \leftarrow x_3 \lor x_4$
- $x_7 \leftarrow x_4 \lor x_5$
- $x_8 \leftarrow x_6 \land x_7$

Formula $\phi$:

- $N(x_4 \iff (x_1 \land x_2)) \land$
- $N(x_5 \iff (\overline{x_1})) \land$
- $N(x_6 \iff (x_3 \lor x_4)) \land$
- $N(x_7 \iff (x_4 \lor x_5)) \land$
- $N(x_8 \iff (x_6 \land x_7)) \land$
- $N(x_8)$
**CLIQUE: An NP-complete graph problem**

Definition:

- Let $G = (V, E)$ be an undirected graph
- A *clique* is a set $C \subseteq V$ such that $(u, v) \in E$ for all $u, v \in C$ such that $u \neq v$

The **CLIQUE** problem:

- **Instance:**
  - A pair $(G, k)$, where $G$ is an undirected graph and $k$ is a positive integer
- **Question:**
  - Is there a clique in $G$ of size $\geq k$?
Proof that \textit{CLIQUE} is NP-complete

- \textit{CLIQUE} \in \textbf{NP}: clear, as the clique itself is a witness

- Need to show \textit{CLIQUE} is NP-hard

- Reduction: 3SAT \leq_P \textit{CLIQUE}

- Let \( \phi \) be a 3-CNF formula:
  \[
  \phi = (a_1 \lor b_1 \lor c_1) \land \cdots \land (a_k \lor b_k \lor c_k)
  \]

- Goal: construct a graph \( G \) such that
  \( \phi \) is satisfiable \iff \( G \) has a clique of size \( k \)
Proof (cont’d):
• $G$ has a $3k$ vertices, one for each literal
• There is an edge between two vertices *unless*
  1. the corresponding literals belong to the same clause
  2. the corresponding literals are *contradictory* (i.e., $x$ and $\overline{x}$)

Example:
$\phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x}_2 \lor x_3) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_3)$
Proof (cont’d):

• Need to show

  \( \phi \) is satisfiable \iff \( G \) has a clique of size \( k \)

• \( \Rightarrow \):

  ◦ suppose \( \phi \) is satisfiable
  ◦ choose a satisfying assignment
  ◦ for each clause, pick one true literal
  ◦ this gives a \( k \)-clique (verify: every pair of vertices is connected)
Proof (cont’d):

• $\leftarrow$:
  
  ◦ suppose $G$ has a $k$-clique $C$
  ◦ by Rule 1, each triple can have at most one element in $C$
  ◦ so $C$ has exactly one element from each triple
  ◦ by Rule 2, we can make a corresponding truth assignment that satisfies $\phi$