**NP Completeness**

Many problems are NP-complete; No polynomial-time solution is known for them, but the existence of such is not known impossible either.

Define problem classes \( \mathcal{P} \) and \( \mathcal{NP} \):

\[
\mathcal{P} = \{ L | \text{some deterministic TM recognizes } L \text{ in time limited by some polynomial}\}
\]

\[
\mathcal{NP} = \{ L | \text{some nondeterministic TM recognizes } L \text{ in time limited by some polynomial}\}
\]

Obviously \( \mathcal{P} \subseteq \mathcal{NP} \), but does it hold that \( \mathcal{P} = \mathcal{NP} \)?

---

If problem \( L \) can be solved in polynomial time, by

1. non-deterministically choosing a candidate solution, and
2. verifying it,

then \( L \) belongs to class \( \mathcal{NP} \)

Example: \( m \times m \) monkey puzzles

---

NP-complete problems are the "most difficult" ones within class \( \mathcal{NP} \)

Comparing difficulty of problems (within polynomial time):

**Polynomial reduction**

Language \( A \) is polynomial-time reducible to \( B \),

\[
A \leq_p B
\]

if we can compute a function \( f \) such that

\[
x \in A \iff f(x) \in B
\]

in polynomial time

---

If \( A \leq_p B \), then "\( w \in A? \)" is (within polynomial time) at most as difficult as the question "\( w \in B? \)".

Let \( f \) be a polynomial reduction \( A \leq_p B \)

\( \sim \) membership test for \( A \):

**procedure** memberOfA(w) returns boolean

\[
w' := f(w);
\]

**return** memberOfB(w');

If \( A \leq_p B \) and

1. \( B \in \mathcal{P} \), then \( A \in \mathcal{P} \);
2. \( A \notin \mathcal{P} \), then \( B \notin \mathcal{P} \).
Definition of NP-completeness

$L$ is **NP-hard** (NP-vaikea), if all languages of $NP$ are polynomially reducible to $L$.

$L$ is **NP-complete** (NP-täydellinen), if (1) $L \in NP$ and (2) $L$ is NP-hard.

The question of efficient solvability of any NP-complete problem captures the famous $P \neq NP$ problem:

**Theorem** Let $L$ be a NP-complete problem. Then $L \in P \iff P = NP$.

**Proof:**

"$\leftarrow$": If $P = NP$, then $L \in P$.

"$\Rightarrow$": Assume $L \in P$. Let $A \in NP$.

Since $A \leq P L$, the question \‘$w \in A$?\' can be solved in polynomial time as \‘$f(w) \in L$?\' \rightarrow $A \in P$.

How do we show problems NP-hard?

By reduction from some NP-hard problem:

(NB direction!)

**Theorem** If $L$ is NP-hard and $L \leq P L'$, then also $L'$ is NP-hard.

**Proof.** Let $A \in NP$, and $f$ reduction for $A \leq P L$ in time $p(n)$.

Let $g$ be reduction for $L \leq P L'$ in time $q(n)$.

Construct $f \circ g$:

\[
\begin{align*}
w' &:= f(w) \\
&\text{return } g(w');
\end{align*}
\]

Now $f \circ g$ is a reduction $A \leq P L'$.

Observations on Satisfiability

**Example** $w = (\neg x \lor y) \land \neg y$.

Value under a given truth assignment is easy to compute.

E.g., $x \leftarrow T, y \leftarrow F$.

Satisfiability is straightforward to check:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$(\neg x \lor y) \land \neg y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

but tedious.
Idea of the proof:

\[ A \in NP \iff A = L(M) \text{ for some NTM } M \text{ with polynomial time compl. } p(n) \]
- using these, construct polynomial reduction

\[ w \mapsto \text{Accept}_A(w), \text{ such that } w \in L(M) \iff \text{Accept}_A(w) \in \text{SAT} \]

First show that we can restrict to 1-tape Turing machines:

**Lemma** Each language of class \( NP \) is accepted by some one-tape NTM in polynomial time

**Proof.** (Sketch)

Let \( M \) be TM with \( k \) tapes. Represent them in TM \( M' \) by a single tape with \( 2k \) “tracks”.
Tracks \( 2i \) correspond to tapes of \( M' \), and tracks \( 2i - 1 \) to positions of tape heads:

\[ M' \text{ simulates a single transition of } M: \]

1. Tape head initially at left, and state corresponds to state \( q \) of \( M \);
2. Scan the tape, moving to state that corresponds to symbols \( a_1, \ldots, a_k \) pointed by simulated tape heads;
3. Choose some transition in accordance to the transition relation \( \delta(q, a_1, \ldots, a_k) \) of \( M \).
4. Return tape head to left; On the way update the symbols at the simulated tape heads

One step of \( M \) is simulated by \( O(p(n)) \) steps of \( M' \)

\[ \sim M' \text{ works in time } O(p(n)^2) \]

Theorem SAT is NP-complete

(S. Cook, 1971)

**Proof.** (Sketch)

1. SAT \( \in NP \): Guess a truth assignment and check whether it satisfies the input formula
2. NP-hardness: Let \( A \in NP \) be recognized with 1-tape NTM \( M, T_M(n) = p(n) \)

Transform \( M \)'s input \( w \) in polynomial time to formula \( \text{Accept}_A(w) \), such that \( w \in A \) iff \( \text{Accept}_A(w) \in \text{SAT} \)

Formula \( \text{Accept}(w) \) describes accepting computations of \( M \) with input \( w \)
Let the states and the tape symbols of $M$ be $\{q_0, \ldots, q_k\}$ and $\{a_0, \ldots, a_m\}$ ($a_0 = b$) 

Use different variables for denoting contents of tape squares, and the state + tape-head position, at different times $0, \ldots, p(n)$:

$$a_i^s.t \iff \text{at time } t, \text{ square } s \text{ contains symbol } a_i,$$

$$q_j^s.t \iff \text{at time } t, \text{ tape-head is at square } s \text{ and } M \text{ is in state } q_j$$

The number of these variables is $O(p(n)^2)$

Accept($w$) can, in poly. time, be simplified into Conjunctional Normal Form (CNF)

$$D_1 \land \cdots \land D_k,$$

where each $D_i$ is a disjunction (a clause) of literals (= variable or its negation)

$\Rightarrow$ CNF-SAT = {satisfiable CNF-formulas} is NP-complete, too.

(Often SAT $\equiv$ CNF-SAT)

Example: $(a \lor b \lor \overline{c}) \land (\overline{a} \lor \overline{b}) \land (b \lor \overline{c}) \land (c)$

Each clause must become true, meaning that

some of its literals must become true

Let's restrict (CNF-)SAT further:

$$3\text{SAT} = \{w \mid w \text{ is satisfiable and in CNF-form, with exactly 3 literals in each clause}\}$$

Example:

$$(a_1 \lor a_2) \land (\overline{a}_1 \lor \overline{a}_2 \lor a_3) \in \text{CNF-SAT} - 3\text{SAT}$$

An equivalent 3CNF-formula:

$$(a_1 \lor a_2 \lor b) \land (a_1 \lor a_2 \lor \overline{b}) \land (\overline{a}_1 \lor \overline{a}_2 \lor a_3)$$
Theorem 3SAT is NP-complete

Proof

(i) 3SAT ∈ \mathcal{NP}: Same as with SAT ∈ \mathcal{NP}

(ii) NP-hardness: CNF-SAT ≤_P 3SAT by transforming too short or too long clauses \( f_i \) of formula \( w = f_1 \land \cdots \land f_m \) as follows:

\[
\begin{align*}
  f_i &= (l_1) \rightarrow (l_1 \lor b_1 \lor b_2) \land (l_1 \lor b_1 \lor \overline{b_2}) \land (l_1 \lor \overline{b_1} \lor b_2) \\
  f_i &= (l_1 \lor l_2) \rightarrow (l_1 \lor l_2 \lor b_1) \land (l_1 \lor l_2 \lor \overline{b_1}) \\
  f_i &= (l_1 \lor l_2 \lor l_3 \lor \cdots \lor l_n) \rightarrow (l_1 \lor l_2 \lor b_2) \land (b_2 \lor l_3 \lor b_3) \land \cdots \land (b_{n-2} \lor l_{n-1} \lor l_n)
\end{align*}
\]

This can be done in linear time, and the new formula is satisfiable iff \( w \) is satisfiable. \( \Box \)