Goal: show 3SAT is $\textbf{NP}$-complete

Need to show $D \leq_{P} 3\text{SAT}$ for all $D \in \textbf{NP}$

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 Memory (RAM)

Input Tape

Control Unit

Output Tape

Random Access Memory
Instruction Set:

- `add 17, 18, 20` # \( m[20] \leftarrow m[17] + m[18] \)
- `sub 17, 18, 20` # \( m[20] \leftarrow m[17] - m[18] \)
- `mul 17, 18, 20` # \( m[20] \leftarrow m[17] \cdot m[18] \)
- `div 17, 18, 20` # \( m[20] \leftarrow \lfloor \frac{m[17]}{m[18]} \rfloor \)
- `ldc 17, 20` # \( m[20] \leftarrow 17 \)
- `lddd 17, 20` # \( m[20] \leftarrow m[17] \)
- `ldi 17, 20` # \( m[20] \leftarrow m[m[17]] \)
- `sti 17, 20` # \( m[m[20]] \leftarrow m[17] \)
- `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` # \( m[20] \leftarrow \) read bit
- `write 17` # write \( m[17] \)
Polynomial time:

\[ n = \text{input length} \]

Requirement: Number of instructions executed 
\[ \leq p(n) \text{ for some polynomial } p \]

Requirement: Number in each memory cell \( \leq p'(n) \)
in absolute value for some polynomial \( p' \)

Implication: highest memory cell addressed is 
\[ \leq p''(n) \text{ for some polynomial } p'' \]
Circuit Satisfiability (CSAT): a first NP-complete problem

Instance:

- A Boolean circuit $C$:
  - inputs $x_1, \ldots, x_n$
  - 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 $x_1, \ldots, x_n$ such that $C(x_1, \ldots, x_n) = 1$?
Linearized representation:

\[ x_4 \leftarrow x_1 \land x_2 \]
\[ x_5 \leftarrow \overline{x_1} \]
\[ x_6 \leftarrow x_3 \lor x_4 \]
\[ x_7 \leftarrow x_4 \lor x_5 \]
\[ x_8 \leftarrow x_6 \land x_7 \]
Proof that CSAT is $\mathbf{NP}$-complete

Need to show that CSAT is $\mathbf{NP}$-hard, i.e., $D \leq_P \text{CSAT}$ for all $D \in \mathbf{NP}$

Let $D \in \mathbf{NP}$

We know there is a poly-time computable function $S$ such that

$$D(x) = 1 \iff S(x, w) = 1 \text{ for some short } w$$

Let $M$ be the polynomial time RAM that computes $S$
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''

\( \alpha_0 \)

\[ m \quad p \quad r \quad y \quad z \]

\( C_{\text{next}} \)

\( \alpha_1 \)

\[ m \quad p \quad r \quad y \quad z \]

\( C_{\text{next}} \)

\[ \vdots \]

\( \alpha_t \)

\[ m \quad p \quad r \quad y \quad z \]

output
Satisfiability (SAT)

Instance:

- A Boolean formula $\phi$:
  - variables $x_1, \ldots, x_n$
  - constants 0, 1
  - operators $\lor$, $\land$, $\neg$
  - Parentheses

Question:

- is there an assignment to the variables $x_1, \ldots, x_n$ such that $\phi(x_1, \ldots, x_n) = 1$?

Formulas are essentially circuits with fan-out restricted to 1
A simple reduction: $\text{CSAT} \leq \text{P SAT}$

- 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</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
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 \( \bar{x} \)

Examples:

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

A special form: 3-CNF

- Each clause consists of 3 distinct literals

The 3SAT problem:

- Instance: a 3-CNF formula
- Question: does it have a satisfying assignment?
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 $\bar{\psi}$
- 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: $CSAT \leq_p 3SAT$
- 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 $\phi$:

- $N(x_4 \iff (x_1 \land x_2)) \land$
- $N(x_5 \iff (\overline{x_1})) \land$
- $N(x_6 \iff (x_3 \lor x_4)) \land$
- $N(x_7 \iff (x_4 \lor x_5)) \land$
- $N(x_8 \iff (x_6 \land x_7)) \land$
- $N(x_8)$
Coping with $\text{NP}$-completeness

Heuristics:
- Very fast $\text{SAT}$-solvers widely used
- Exponential worst-case running time
- Much faster for many “typical” inputs

Approximation algorithms:
- Example: compilers use heuristic algorithms for graph coloring to do register allocation
- Results may not be optimal, but “close enough”

Still an active area of research:
- Proving $\text{NP}$-completeness is just the first step
- This tells us we should focus on finding good heuristic and/or approximation algorithms