Reduction #3: Satisfiability

- Let $X = \{x_1, \ldots, x_n\}$ be boolean variables
  - A term or literal is $x_i$ or $\neg x_i$.
  - A clause is or of several terms $(t_1 \lor t_2 \lor \ldots \lor t_k)$.
  - A formula is and of several clauses
  - An assignment $\phi : X \rightarrow \{0, 1\}$ gives T/F to each variable.
  - $\phi$ satisfies formula if all clauses evaluate to True.

**Example.**

\[(x_1 \lor \neg x_2) \land (x_1 \lor x_4 \lor \neg x_3) \land (\neg x_1 \lor x_4) \land (x_3 \lor x_2)\]

**Reduction**

\[(x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3)\]

- Associate nodes in graph with literals ($\geq 2$ per variable).
- Associate $3$ nodes per clause in a gadget.
- If $\phi(x_i) = 1$ in assignment, then cannot select some nodes.

Formally

- **Given** $(x_1, \ldots, x_n)$ and clauses $C_1, \ldots, C_m$.
- **Make graph** with:
  - Vertices $v_{i0}$ and $t_{ij}$ for $i \in [n], j \in [m]$.
  - Edges $(v_{i0}, v_{ij})$ for $i$ and $(t_{jk}, t_{jk'})$ for $k, k' \in [3]$.
  - If $j$th clause is $x_a \lor \neg x_b \lor x_c$, edges $(t_{j1}, v_{a0}), (t_{j2}, v_{b0}), (t_{j3}, v_{c0})$.
- If $G$ has IS of size $n + m$, output True, else False.

Recap

- Reductions. $Y \leq_P X$ if can solve $Y$ in poly-time with algorithm for $X$.
- New problems. INDEPENDENTSET, VERTEXCOVER, SETCOVER, SAT, 3-SAT.
- Results.

\[3\text{-SAT} \leq_P \text{IS} \leq_P \text{VC} \leq_P \text{SC}\]

\[\text{VC} \leq_P \text{IS}\]
### Satisfiability Proof

\[
(x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3)
\]

**Claim.** Reduction takes polynomial time.

**Claim.** Graph has IS of size \(n + m\) if and only if formula satisfiable.

### 3-SAT Reduction

**Theorem.** \(3\text{-SAT} \leq_p \text{IndependentSet}\)

- For every 3-SAT formula, exists a graph \(G\) s.t. formula satisfiable if and only if \(G\) has IS of size \(n + m\).
- Does not imply \(\text{IndependentSet} \leq_p \text{3-SAT}\).
- For this, need to prove: For every \((G, k)\), exists formula that is satisfiable iff \(G\) has IS of size \(k\).

### Certification and NP.

- Algorithm \(A\) solves problem \(X\) if \(A(s) = \text{TRUE}\) iff \(s \in X\).
- Running time now measured in \(|s|\), still want polytime.
- \(\mathcal{P}\): problems that can be solved by a polytime algorithm.
- \(B\) is a polytime **certifier** for problem \(X\) if
  - \(B\) is a polytime algorithm of two inputs \(s, t\).
  - \(s \in X\) iff exists \(t\) with \(|t| \leq \text{poly}(|s|)\) and \(B(s, t) = \text{TRUE}\).
- **Example.** Certifier for independent set.
- \(\mathcal{NP}\): problems with polytime certifier.

### A class of problems

- Decision vs certification.
- Seems hard to find a large independent set.
  - Or check if one exists.
- But easy to certify a proposed solution, by checking for adjacent vertices.
- Formal languages and decision problems.
  - Encode problem inputs as binary strings \(s\).
  - A decision problem \(X\) is the set of binary strings that have \(\text{TRUE}\) answer.
  - Algorithm \(A\) solves problem \(X\) if \(A(s) = \text{TRUE}\) iff \(s \in X\).

### P and NP

**Claim.** \(\mathcal{P} \subseteq \mathcal{NP}\).

**Proof.**

- If \(X \in \mathcal{P}\), exists algorithm \(A\) that solves \(X\).
- Need to design certifier \(B\).
  - Set \(B(s, t) = A(s)\).
  - \(B\) runs in polynomial time
  - If \(s \in X\), \(B(s, t) = A(s) = \text{TRUE}\) for all \(t\).
  - If \(s \notin X\), \(B(s, t) = A(s) = \text{FALSE}\) for all \(t\).
Some NP problems.

- **IndependentSet**
- **VertexCover**
- **SetCover**
  - Basically all problems we have seen so far!
- **Unsatisfiability** — not in $\mathcal{NP}$.

Million dollar question

**Question.** Does $\mathcal{P} = \mathcal{NP}$?

Can make some progress by considering “hardest” $\mathcal{NP}$ problems.

**Definition.** $X$ is NP-Complete if $X \in \mathcal{NP}$ and for all $Y \in \mathcal{NP}$ $Y \leq_P X$.

- If $X$ is NP-Complete then $X$ has poly-time algorithm iff $\mathcal{P} = \mathcal{NP}$.

Circuit-SAT

**Problem.** Given a boolean circuit with some inputs and single boolean output, are there inputs that produce 1 at the output?

- A circuit is a labeled DAG.
- Sources (no incoming edges) labeled with constant or with input variable name.
- Other nodes labeled with $\land$ (and), $\lor$ (or), $\neg$ (not).
- Single node with no outgoing edges computes the output bit.

**Theorem.** Circuit-SAT is NP-Complete.

**Proof (Idea).**

- A poly-time algorithm once input length is fixed can be executed on a poly-sized circuit.
- Not surprising since our hardware is circuits!
- Need to show that arbitrary $X \in \mathcal{NP}$ has $X \leq_P \text{Circuit-SAT}$.
- All we know about $X$ is its efficient certifier $B(\cdot, \cdot)$.
- Encode $B(s, \cdot)$ as a circuit with $\text{poly}(|s|)$ inputs.
- Satisfiable iff exists $t$ with $|t| \leq \text{poly}(|s|)$ s.t. $B(s, t) = \text{true}$ if $s \in X$.

Back to 3-SAT

**Claim.** If $Y$ is NP-complete and $Y \leq_P X$, then $X$ is NP-complete.

**Theorem.** 3-SAT is NP-Complete.

- Clearly in $\mathcal{NP}$.
- Prove by reduction from CircuitSAT.

**Example.**
The Reduction

- One variable $x_v$ per circuit node $v$.
- Clauses to enforce circuit computations.
  - If $v$ is $\neg$ then $v$ has one input $u$ and can add clauses $(x_v \lor x_u), (\neg x_v \lor \neg x_u)$.
  - If $v$ is $\lor$ with $u, w$ incoming then $(x_v \lor \neg x_u), (x_v \lor \neg x_w), (\neg x_v \lor x_u \lor x_w)$.
  - If $v$ is $\land$ then $(\neg x_v \lor x_u), (\neg x_v \lor x_w), (x_v \lor \neg x_u \lor \neg x_w)$.
- Input bits get set with $(x_v)$ if fixed to one and $(\neg x_v)$ otherwise.
- Clause $(x_o)$ for output bit.

Final steps

- This formula satisfiable iff circuit is satisfiable.
- But not a 3-sat formula! It has clauses of size 1 and 2.
  - Fix: 4 new variables $z_1, \ldots, z_4$ where $z_1, z_2$ forced to be 0.
  - Include those two in any short clause.

Theorem. INDEPENDENTSET, VERTEXCOVER, SETCOVER, SAT, 3-SAT are all NP-Complete.