Reducing one (decision) problem to another

\[ \text{input for } A \rightarrow \text{Transform to input/problem } B \rightarrow \text{solve } B \rightarrow \text{output of } B \rightarrow \text{output of } A \rightarrow \text{yes/no} \]

Solve A using B: \( T_R + T_B \)

Is L.I.S.\((A) \geq k\)?

\[ \text{copy } A \rightarrow B \rightarrow \text{sort } B \rightarrow \text{is L.C.S. } (A,B) \geq k? \rightarrow \text{yes/no} \]

Yes/No
Lower bound for one (decision) problem via another

\[ T_R = o(f(n)) \]

solve D: known \( \Omega(f(n)) = T_R + T_C \Rightarrow T_C = \Omega(f(n)) \)

- quick transformation \( \Rightarrow \) C is at least as difficult as D
  (fast solution for C \( \Rightarrow \) fast solution for D)
NPC REDUCTION FOR ONE (DECISION) PROBLEM VIA ANOTHER

LIKE ESTABLISHING AN ALMOST CERTAIN LOWER BOUND FOR C.

\[
\text{input for } D \quad \rightarrow \quad \text{Transform to input/problem } C \quad \rightarrow \quad \text{solve } C \quad \rightarrow \quad \text{output of } C \quad \rightarrow \quad \text{yes} \quad \text{yes} \quad \text{no} \quad \text{no}
\]

\[ T_R \text{= polynomial} \]

\[ \text{solve } D: \text{ known NPC} = T_R + T_c = \text{poly-time } + T_c \]

\[ \Rightarrow T_c \text{ is NP-hard} \]

& if C is in NP then C is NPC

(or \( T_c \text{ is polynomial} \Rightarrow P=NP \))
$A \leq_p B$ : in polynomial time, $A$ can be transformed to $B$

$\Rightarrow B$ is at least as hard as $A$.

Solving B in poly-time $\Rightarrow$ solving A in poly-time

If $\{\text{any problem in NP}\} \leq_p B$ then B is NPC.

Notice if $A \& B$ are NPC then $A \leq_p B \& B \leq_p A$.

$\sim A \not{\leq_p} B$
Thousands of NPC Problems

Reductions/transformations between them resemble a di-graph.

\( \geq \) a DAG? No.

\#strongly connected components? Must be 1, precisely.
THOUSANDS OF NPC PROBLEMS

Reductions/transformations between them resemble a di-graph.

\( \subseteq \text{a DAG? No.} \)

#strongly connected components?
Must be 1. precisely.

FACT
\( \exists \) at least one problem A s.t.
A \( \geq p \) \{everything\}
\& \{everything\} \( \geq p \ldots \geq p \ldots \geq p \) A
Circuit SAT (satisfiability)
The first NPC problem.
Given a circuit, can the output ever be 1?

\[
\begin{align*}
&x_i: \text{input} \\
&\begin{cases}
0 & x_1 \\
1 & x_2 \\
0 & x_3 \\
1 & x_4 \\
0 & x_5
\end{cases}
\end{align*}
\]
Circuit SAT (satisfiability)

The first NPC problem.
Given a circuit, can the output ever be 1?
1 - in NP ... intuitive
2 - every problem can be described as a circuit
\{everything\} \leq_p \text{circuit-SAT}

...and we don't know how to solve this in polynomial time. (not in P)
Given a circuit-SAT instance, transform (quickly) into a Boolean SAT.

by solving the Boolean SAT you get an answer for circuit SAT

\[ w \land (w \leftrightarrow (y_4 \land y_2 \land y_3)) \land (y_4 \leftrightarrow (y_4 \lor y_2)) \land (y_1 \leftrightarrow (c_1 \lor c_2)) \land (y_2 \leftrightarrow \lnot c_3) \land (y_3 \leftrightarrow (c_4 \land c_5)) \]
3-SAT

\( (x_1 \lor x_2 \lor x_3) \)
\( \land (x_1 \lor x_4 \lor x_5) \)
\( \land (x_2 \lor \overline{x}_4 \lor x_{13}) \)
\( \land (x_2 \lor \overline{x}_3 \lor x_3) \)

\( \vdots \)

\( n \) clauses

each with 3 literals
\[ (a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d}) \]

\( k \) clauses

**3-SAT** → **boolean SAT** → **circuit SAT**

**Construct graph (quickly)**

for every triple \( (x \lor y \lor z) \) make a triple of vertices, linked together
\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\]  
\{k \text{ clauses}\}

constructed graph (quickly)

link every vertex to any other vertex that has the same variable but negated
\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\] 

\[k \text{ clauses}\]

**3-SAT** ➔ **boolean SAT** ➔ **Circuit SAT**

**Ask:** does this graph have \(k\) independent vertices?

To find \(k\) independent, you must use each triple, because each triple can only have 1 independent vertex.

As soon as you select a vertex in a triple, that blocks you from using its negation anywhere else.
\[(a \lor b \lor c) \land (b \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor c \lor d) \land (a \lor \bar{b} \lor \bar{d})\] \{ k clauses \}

- If you can't find k independent vertices, then some triple had no selectable vertex... i.e. all 3 were blocked by selections in other triples. That corresponds to an unsatisfiable row in 3-SAT.

- Ask: does this graph have k independent vertices?
  - If yes, then 3-SAT is \(\checkmark\)
  - If no, then 3-SAT is \(\times\)
\((a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor \overline{d}) \land (a \lor \overline{b} \lor \overline{d})\) 

\(k\) clauses

3-SAT ➔ boolean SAT ➔ Circuit SAT

Independent set

Construct graph (quickly)

\(\leq 1\) per triplet

Ask: does this graph have \(k\) independent vertices?

If yes, then 3-SAT is \(\checkmark\)
If no, then 3-SAT is \(\times\)
CLIQUE in a graph: subset of $V$ s.t. all pairs of vertices share edges.

independent set size $k$ $\iff$ clique size $k$

G $\xrightarrow{\text{transform}}$ complement $(G)$

Asking if a graph has a clique of size $k$: NPC
OTHER HARD PROBLEMS

- knapsack: given item types, w/ size & value, fill a bag w/ max value (multiples OK)
- subset sum: does a subset of given integers sum to t?
- partition: split set of integers in 2 groups with equal sums
- planar SAT: SAT without wire crossings
- set cover: given some sets, select min# sets s.t all elements are present
- hitting set: given sets, select min# elements s.t. all sets are represented
- longest path: visiting each vertex once
- Steiner tree: ~MST on selected subset of a graph
- traveling salesman: min-cost simple cycle on weighted complete graph
OTHER HARD PROBLEMS

... & some are even harder

Tetris    Minesweeper    Lemmings

Mario Bros    Pac-man

Prince of Persia    Portal    Doom

etc