Recap

- L is in **NP** means:
  There is a language L' in **P** and a polynomial p so that
  $\forall x : x \in L \Leftrightarrow [\exists y \in \{0,1\}^* : |y| \leq p(|x|) \wedge \langle x, y \rangle \in L']$

- $L_1 \leq L_2$ means:
  For some polynomial time computable map r:
  $$\forall x : x \in L_1 \Leftrightarrow r(x) \in L_2$$

- L is **NP**-hard means:
  $$\forall L' \in \mathbf{NP} : L' \leq L$$

- L is in **NPC** means:
  $L \in$ **NP** and L is **NP**-hard

How to establish NP-hardness

Lemma

If L_1 is NP-hard and $L_1 \leq L_2$, then L_2 is NP-hard

SAT and close relatives

SAT

- Given: CNF formula F on n variables.
- Question: Does there exist $x \in \{0,1\}^n$ such that $F(x) = 1$?

CircuitSAT

- Given: Boolean Circuit C on n variables.
- Question: Does there exist $x \in \{0,1\}^n$ such that C$(x) = 1$?

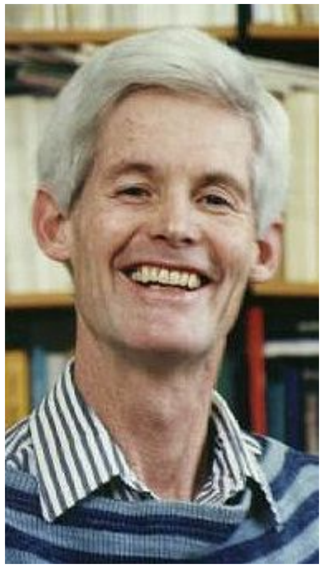
kSAT

- Given: kCNF formula F on n variables.
- Question: Does there exist $x \in \{0,1\}^n$ such that $F(x) = 1$?

- A kCNF formula is a CNF formula where each clause has at most k literals.

SAT

- SAT is in NP.
- Cook's theorem (1972): SAT is NP-hard.



3SAT is NP-complete

Recall reduction CircuitSAT $\leq$ SAT (Tseitin transformation):

Let C be a Boolean circuit on n variables.

Construct CNF formula F with

- Variables: One variable g for every gate g of C.
- Clauses:
  - For each gate of C, clauses that express the computation of the gate. E.g., $g \Leftrightarrow h_1 \wedge h_2$ expresses that gate g is the Boolean conjunction of gates h_1 and h_2. For every gate this is a Boolean function on at most 3 variables, which can be expressed as a CNF formula.
  $$(\neg g \vee h_1) \wedge (\neg g \vee h_2) \wedge (g \vee \neg h_1 \vee \neg h_2)$$
  - For the output gate g_{out} of C, the unit clause (g_{out}).

SAT $\leq$ 3SAT

Construct $r(F) \rightarrow F'$ such that $F \in SAT \Leftrightarrow F' \in 3SAT$

<u>Idea:</u> Break up clauses $(\ell_1 \vee \cdots \vee \ell_k)$ for $k > 3$ into smaller clauses

Example: $(\ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4)$

Can be generalized for arbitrary $(\ell_1 \vee \cdots \vee \ell_k)$

Resolution

Let F be the following CNF formula:

$$(x \vee P_1) \wedge (x \vee P_2) \wedge \cdots \wedge (x \vee P_k) \wedge$$
$$(\neg x \vee Q_1) \wedge (\neg x \vee Q_2) \wedge \cdots \wedge (\neg x \vee Q_\ell) \wedge$$
$$R$$

where R does not contain the variable x.

Then $\text{Resolve}(F, x)$ is defined to be the following CNF formula:

$$(P_1 \vee Q_1) \wedge (P_2 \vee Q_1) \wedge \cdots \wedge (P_k \vee Q_1) \wedge$$
$$(P_1 \vee Q_2) \wedge (P_2 \vee Q_2) \wedge \cdots \wedge (P_k \vee Q_2) \wedge$$
$$\ldots$$
$$(P_1 \vee Q_\ell) \wedge (P_2 \vee Q_\ell) \wedge \cdots \wedge (P_k \vee Q_\ell) \wedge$$
$$R$$

The Davis-Putnam procedure

Input: CNF formula F

Output: Satisfiability of F

while F is not empty **do**:

    if F contains an empty clause **then**:

        return false

    let $x \in \mathrm{vars}(F)$

    let $F \coloneqq \mathrm{resolve}(F, x)$

return true

The implication graph of a 2CNF

$$(x \vee y) \equiv (\neg x \Rightarrow y) \equiv (x \Leftarrow \neg y)$$

Let F be a 2CNF formula. Define the directed graph $G(F)$:

- The nodes of $G(F)$ are variables of F and their negations (i.e. all possible literals)
- The arcs of $G(F)$ represent the two equivalent implications of every clause of F.

2SAT and connectivity

Theorem

F is satisfiable if and only if:

there is no variable x such that there is *both* a path from x to $\neg x$ *and* from $\neg x$ to x in $G(F)$

Finding a satisfying assignment

Input: 2CNF formula *F*

Output: Satisfiability of *F*

$S =$ empty set (Set of "fixed" variables.)

while $S \neq \mathrm{vars}(F)$, **do**:

Pick x in $\mathrm{vars}(F)$ but not in S. Add x to S.

If there is a path in $G(F)$ from x to $\neg x$, then set $x = 0$, else set $x = 1$.

If literal ℓ is set to 1, then explore the graph $G(F)$ from ℓ. Set all unset literals ℓ' reachable from ℓ to 1.