SAT problem.

Boolean variables $\{x_1, \dots x_n\}$ $x_i \in \{0,1\}$

Formula in CNF over variables: conjunction of clauses; clause is a disjunction of literals; literal either variable x_i or negation $\neg x_i$ / $\bar{x}_i$ / $\sim x_i$.

$F = \{y \vee z,\ y \vee \bar{z},\ x \vee \bar{y} \vee z,\ x \vee \bar{y} \vee \bar{z}\}$. set all vars to true.

Goal: find 0/1 assignment to variables st every clause is satisfied.

$G = \{y \vee z,\ y \vee \bar{z},\ x \vee \bar{y} \vee z,\ x \vee \bar{y} \vee \bar{z},\ \bar{x} \vee \bar{y}\}$. not satisfiable.

Brute-force algorithm:
for $\alpha \in \{0,1\}^n$
  test if $F\restriction\alpha$ is satisfied

runtime: $2^n \cdot m \cdot n$.

Heuristic algorithms.

DPLL algorithm (Davis, Putnam, Logemann, Loveland '60s)

smart backtracking

$F = \{y \vee z,\ y \vee \bar{z},\ x \vee \bar{y} \vee z,\ x \vee \bar{y} \vee \bar{z}\}$.

what if we set $y = 0$?
$z = 1$.

$\cancel{y} \vee z$

state: partial assignment $\alpha \subseteq \{0,1,*\}^n$ ← not assigned yet.
+ decision / forced choice.

$\alpha = *^n$
while not solved:
  if $F\restriction\alpha$ is contradictory: backtrack().
  if $F\restriction\alpha$ has unit clause: set $x_i = b$ ← x_i^b
  otherwise $x_i = b$ (decision)

$x_i^0 = \bar{x}_i$
$x_i^1 = x_i$

backtrack: find first decision, flip value, mark as forced. → starting from back.

$\alpha = \langle x_{i_1} \overset{d}{=} b_1,\ x_{i_2} \overset{f}{=} b_2,\ \dots \rangle$

$F = \{y \vee z,\ y \vee \bar{z},\ x \vee \bar{y} \vee z,\ x \vee \bar{y} \vee \bar{z}\}$.

$\overset{d}{x=0}$. $\overset{d}{y=0}$, $\overset{f}{z=1}$

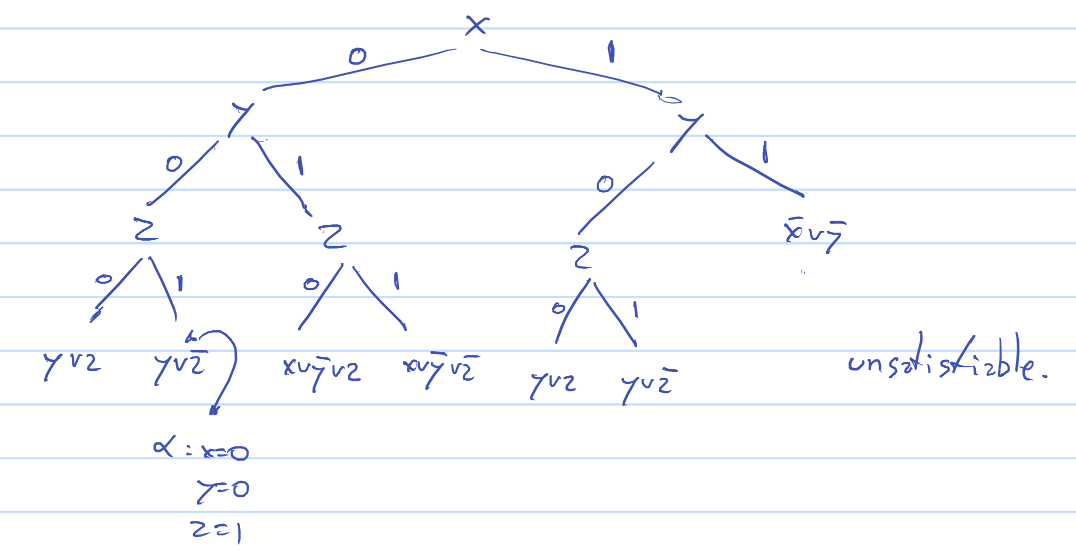
$\overset{d}{x=0}$ $\overset{f}{y=1}$, $\overset{f}{z=1}$

$\overset{f}{x=1}$ $\overset{d}{y=1}$

$x=1$, $y=1$, $z=*$
is a satisfying assignment.

→ $y \vee z$, $y \vee \bar{z}$, $\bar{y} \vee z$, $\bar{y} \vee \bar{z}$.
z, $\bar{z}$, /, /
/ !!
/ / z, $\bar{z}$
/ !!
$y \vee z$ $y \vee \bar{z}$ / /
/ /

$G = \{y \vee z,\ y \vee \bar{z},\ x \vee \bar{y} \vee z,\ x \vee \bar{y} \vee \bar{z},\ \bar{x} \vee \bar{y}\}$.



unsatisfiable.

Running time:
if F is sat and we are lucky: running time is $n \cdot poly(n)$
on unsatisfiable formulas: running time $\geqslant$ size of search tree.

worst case: search tree is $\leq 2^n$
$\exists$ formulas s.t. is $\geqslant 2^{\Omega(n)}$
$2^{n/4}$

Resolution proofs.

Resolution rule

$$\frac{x \vee y \vee z \qquad y \vee \bar{z} \vee t}{x \vee y \vee t.}$$

$z=0$: ⇒ $x \vee y$

$z=1$ ⇒ $y \vee t$

$$\frac{C \vee x \qquad D \vee \bar{x}}{C \vee D.}$$

$z=0 \vee z=1$ ⇒ $(x \vee y) \vee (y \vee t)$
$x \vee y \vee t$.

Resolution proof of unsatisfiability. / resolution refutation / of formula F:

sequence of lines $C_1 \dots C_L$ each line is a clause. either from F or by applying resolution rule.

$C_L = !!$ (clause without any literals).

$G = \{y \vee z,\ y \vee \bar{z},\ x \vee \bar{y} \vee z,\ x \vee \bar{y} \vee \bar{z},\ \bar{x} \vee \bar{y}\}$.

$(y \vee z,\ y \vee \bar{z},\ y,\ x \vee \bar{y} \vee z,\ x \vee \bar{y} \vee \bar{z},\ x \vee \bar{y},\ \bar{x} \vee \bar{y},\ \bar{y},\ !!)$

$x \vee \bar{y}$
$\bar{y}$

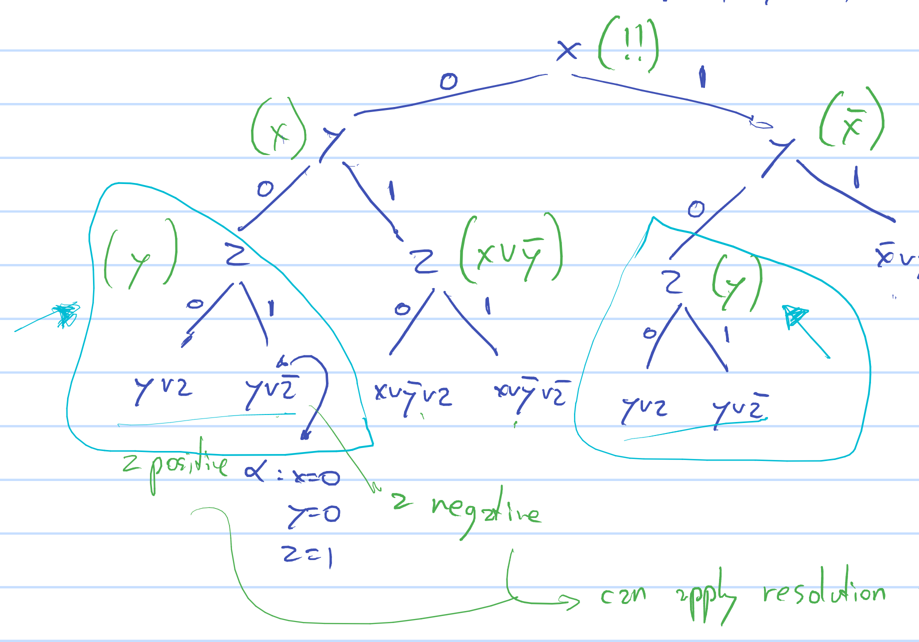
thm: resolution is sound and complete
↳ no satisf. formula has a resolution proof.
↳ every unsat formula has a resolution proof.

worst case: length is $\leq n \cdot 2^n$

thm: (Haken '85) exist family of formulas on n variables, poly(n) clauses st shortest resolution proof of F_n has length $2^{n^{\Omega(1)}}$.

(Urquhart '88) $2^{\Omega(n)}$.

$G = \{y \vee z,\ y \vee \bar{z},\ x \vee \bar{y} \vee z,\ x \vee \bar{y} \vee \bar{z},\ \bar{x} \vee \bar{y}\}$.



unsatisfiable.

lemma if search tree of size L then resolution proof of size L.

cor: $\exists$ formulas that require $2^{\Omega(n)}$ to solve with DPLL.

def tree-like resolution proof: res. proof st graph is a tree
↳ each vertex is a clause of proof
edge between C and D if
→ C is used to derive D.

$$\frac{C \qquad E}{D}$$

proof is tree-like if each clause only used once.

How to avoid reusing work? with CDCL algorithm. ('99)
↓
conflict-driven clause learning.

CDCL α ← assignment $\mathbb{D}$ ← database of learned clauses.

while not solved:
  if conflict: backtrack. ← learn()
  if unit : force value.
  o/w decision