Given a formula F, can we determine whether it is satisfiable

Let F is over X variables, where $X = \{x_1, \dots, x_n\}$

i.e $|\text{variables}(F)| = n$

checkSAT(F)

---

for τ in 2^n do
    if $F(\tau) = 1$, then
        return SAT, τ
return UNSAT

checkSAT(F)

for τ in 2^n do
    if $F(\tau) = 1$, then
        return SAT, τ
return UNSAT

find an satisfying assignment of F, can take exponential time

Can we do better ?

→ we don't know !!

Resolution

$$(p \vee \alpha) \wedge (\neg p \vee \beta) \iff (\alpha \vee \beta)$$

$F = (p \vee \alpha) \wedge (\neg p \vee \beta)$
$G = (\alpha \vee \beta)$

$\Big\}$ F & G are equisatisfiable formulas

Resolution Refutation

list of clauses $C_1, C_2, \ldots, C_t$ is a resolution refutation of formula F_{CNF} if:

i) C_t is empty $\Box$
ii) $C_k \in F_{CNF}$ or C_k is derived using resolution from C_i, C_j, where $i, j < k$.

Resolution Refutation

list of clauses $C_1, C_2, \ldots, C_t$ is a resolution refutation of formula F_{CNF} if:

i) C_t is empty $\square$
ii) $C_k \in F_{CNF}$ or C_k is derived using resolution from c_i, c_j, where $i,j < k$.

example

$c_i = P \vee \alpha$
$c_j = \neg p \vee B$
$C_K = \alpha \vee B$

C_k is derived from c_i & c_j

models $(F) = 0$
F is UNSAT

Example:

$$F = \underbrace{(\neg p \vee \neg q \vee r)}_{C_1} \wedge \underbrace{(\neg p \vee q)}_{C_2} \wedge \underbrace{p}_{C_3} \wedge \underbrace{\neg r}_{C_4}$$

Resolution on C_1 & C_3 : $\dfrac{(\neg p \vee \neg q \vee r) \wedge (p)}{C_5 : \neg q \vee r}$

Resolution on C_2 & C_3 : $\dfrac{(\neg p \vee q) \wedge p}{C_6 : q}$

Resolution on C_5 & C_4 : $\dfrac{(\neg q \vee r) \wedge (\neg r)}{C_7 : \neg q}$

Resolution on C_6 & C_7 : $\dfrac{q \wedge \neg q}{C_8 : \square}$] – Refutation

list of clauses $C_1, C_2, \ldots, C_8$ is a resolution refutation of F

<u>Thm</u>: A formula F_{CNF} is refutable if and only if F_{CNF} is unsatisfiable.

$\rightarrow$ direction is easy to see: if a formula F_{CNF} is refutable then F_{CNF} is unsatisfiable.

$\leftarrow$ direction: if F_{CNF} is unsatisfiable then F_{CNF} is refutable

└ Homework: Induction on # of propositional variables.

⊙ Use resolution Refutation to prove the validity of a formula:

To prove: F is valid

1. convert F to $(\neg F)$
2. convert $(\neg F)$ to CNF, say F'_{CNF}
3. Find a resolution refutation of F'_{CNF}
4. if F''_{CNF} is refutable, then F is valid

Bottleneck of Resolution:

→ Space required to perform resolution

- At every resolution step: ${}^{n}C_2$ $n = |\text{clauses}|$
  new clauses is added to the set of clauses.
- this is done linear # of times, hence overgrowth can be exponential

Resolution is EXP SPACE

Davis-Putnam (DP '1960) Algorithm

Martin Davis & Hil

1. Start with F_{CNF}
2. preform Resolution Untill
   - → empty clause is derived
     - ↳ return UNSAT
   - → No further resolution is possiable
     - ↳ return SAT

One of those two case will occur becoz resolution refutation is sound & complete

Davis-Putnam (DP' 1960) Algorithm

Pick a literal l that occurs with both polarities in F.

for every clause c in F containing l and every clause c' in F containing its negation $\neg l$ do

Resolve c & c'

$r \leftarrow (c \setminus \{l\}) \cup (c' \setminus \{\neg l\})$

$F \leftarrow$ add_to_formula(r, F)

for every clauses c that contain l or $\neg l$ do

$F \leftarrow$ remove-from-formula(c, F)

Davis-Putnam (DP '1960) Algorithm

if F has empty clause then
return UNSAT

Pick a literal l that occurs with both polarities in F.

for every clause c in F containing l and every clause c' in F containing its negation $\neg l$ do

Resolve c & c'

$r \leftarrow (c \setminus \{l\}) \cup (c' \setminus \{\neg l\})$

$F \leftarrow$ add_to_formula(r, F)

for every clauses c that contain l or $\neg l$ do

$F \leftarrow$ remove-from-formula(c, F).

Davis-Putnam (DP '1960) Algorithm

if F has empty clause then
    return UNSAT
if $\nexists\, l$ that occur with both polarities in F,
    return SAT

Pick a literal l that occurs with both polarities in F.

for every clause c in F containing l and every clause c' in F containing its negation $\neg l$ do
    Resolve c & c'
    $r \leftarrow (c \setminus \{l\}) \cup (c' \setminus \{\neg l\})$
    $F \leftarrow$ add_to_formula(r, F)

for every clauses c that contain l or $\neg l$ do
    $F \leftarrow$ remove-from-formula(c, F).

Davis-Putnam (DP '1960) Algorithm

if F has empty clause then
    return UNSAT
if $\nexists$ ℓ that occur with both polarities in F,
    return SAT

is this correct?
↳ does it terminate when $F = P \vee \neg P$

Pick a literal ℓ that occurs with both polarities in F.

for every clause c in F containing ℓ and every clause c' in F containing its negation $\neg \ell$ do
    Resolve c & c'
    $r \leftarrow (c \setminus \{\ell\}) \cup (c' \setminus \{\neg \ell\})$
    $F \leftarrow$ add_to_formula(r, F)

for every clauses c that contain ℓ or $\neg \ell$ do
    $F \leftarrow$ remove-from-formula(c, F).

Davis-Putnam (DP'1960) Algorithm

DP(F)
{

For every clause c in F that contains both l and $\neg l$ do
  $F \leftarrow$ remove-from-formula(c, F)

stopping conditions

if F is empty then
  return SAT

if F has empty clause then
  return UNSAT

if $\nexists$ l that occur with both polarities in F
  return SAT

Can we make it better?

Pick a literal l that occurs with both polarities in F.

for every clause c in F containing l and every clause c' in F containing its negation $\neg l$ do
  Resolve c & c'
  $r \leftarrow (c \setminus \{l\}) \cup (c' \setminus \{\neg l\})$
  $F \leftarrow$ add_to_formula(r, F)

for every clauses c that contain l or $\neg l$ do
  $F \leftarrow$ remove-from-formula(c, F).

DP(F)

}

Pure literal elimination :-

Pure literal : a literal ℓ all of which occurrences in F have the same polarity

Example:

$F = (p \vee q \vee r) \wedge (\neg q \vee r) \wedge (p \vee \neg r) \wedge (p \vee \neg q)$

→ literal p has positive polarity in all occurrence in F

→ p is pure literal

$F = (p \vee \neg q \vee r) \wedge (\neg q \vee r) \wedge (\neg p \vee \neg r) \wedge (p \vee \neg q)$

→ literal q has negative polarity in all occurrence in F

→ q is pure literal

Pure literal elimination :-

Pure literal : a literal l all of which occurrences in F have the same polarity

for every clause that contains a pure literal:
F ← remove-from-formula(F)

→ pure literal can be set as per polarity to satisfy the clause.

Davis-Putnam (DP '1960) Algorithm

DP(F)
{

For every clause c in F that contains both ℓ and $\neg\ell$ do
  F ← remove-from-formula(c, F)

while there is a pure literal ℓ do
  for every clause c that contains ℓ do
    F ← remove_from_formula(c, F)

# stopping conditions
if F is empty then
  return SAT
if F has empty clause then
  return UNSAT

Pick a literal ℓ that occurs with both polarities in F.

for every clause c in F containing ℓ and every clause c' in F containing its negation $\neg\ell$ do
  Resolve c & c'
  $r \leftarrow (c \setminus \{\ell\}) \cup (c' \setminus \{\neg\ell\})$
  F ← add_to_formula(r, F)

for every clauses c that contain ℓ or $\neg\ell$ do
  F ← remove-from-formula(c, F).

DP(F)

}