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)

while there is a pure literal l do
for every clause c that contains 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

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)

}

Show run of DP on F:

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

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

* No pure literal, No clause with $(\ell \vee \neg \ell)$

Pick p →

$(p \vee q \vee r) \wedge (q \vee \neg r \vee \neg s) \wedge (\neg q \vee s) \wedge (\neg p \vee \neg s)$

$(q \vee r \vee \neg s) \wedge (q \vee \neg r \vee \neg s) \wedge (\neg q \vee s)$

Pick q →

$(r \vee \neg s \vee s) \wedge (\neg r \vee \neg s \vee s)$

clause with $(\ell \vee \neg \ell)$ – remove from formula

F is empty

↓

<u>SAT</u>

Show run of DP algorithm on :

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

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

* No pure literal, no clause with $l \vee \neg l$

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

pick p

$(q \vee r) \wedge (q \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg q \vee \neg r)$

pick q

$(r) \qquad (r \vee \neg r) \qquad (\neg r \vee r) \qquad (\neg r)$

clause with $l \vee \neg l$ → remove from formula

$(r) \qquad \neg(r)$

pick r

$\square$ ← formula has empty clause - UNSAT

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

Any optimization?

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

1.

if $\neg p$ is true, (i.e $p = 0$), then $(\neg p \vee r) \wedge (\neg p \vee \neg r)$ is true

→ Remove c that has $\neg p$ in it.

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

1. if $\neg p$ is true, (i.e $p = 0$), then $(\neg p \vee r) \wedge (\neg p \vee \neg r)$ is true

→ Remove c that has $\neg p$ in it.

2. if $\neg p$ is true (i.e. $p = 0$), then $(p \vee q)$ is equisatisfiable with q

if $\neg p$ is true, i.e. $p = 0$, then $(p \vee \neg q)$ is equisatisfiable with $\neg q$

- Add $c / \{\neg p\}$ to formula
- Remove c. from the formula.

Unit Propogation (F)

```
{
While F contains a unit clause (l) do

    for every clause c in F that has l do:
        F ← remove_from_formula(c, F)

    for every clause c in F that has ¬l do:
        F ← remove_from_formula(c, F)
        F ← add_to_formula(c/{¬l}, F)
}
```

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)

F ← unit propagation(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)

}

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)

F ← unit propagation(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)

}

DP procedere

Analysis of DP procedure

The resolution step (DP procedure) leads to a worst-case exponential blow-up in the size of formula.

↳ say n clause, l literal is positive in $n/2$ clauses & negative in remaining $n/2$ clauses

↳ how many additional clauses after picking l in DP procedure?

Davis - Putnam - Logemann-Loveland (DPLL '62) Algorithm:

→ Complete & Sound Algorithm & takes linear Space in the worst case.

→ Still the basis of SAT Solvers.

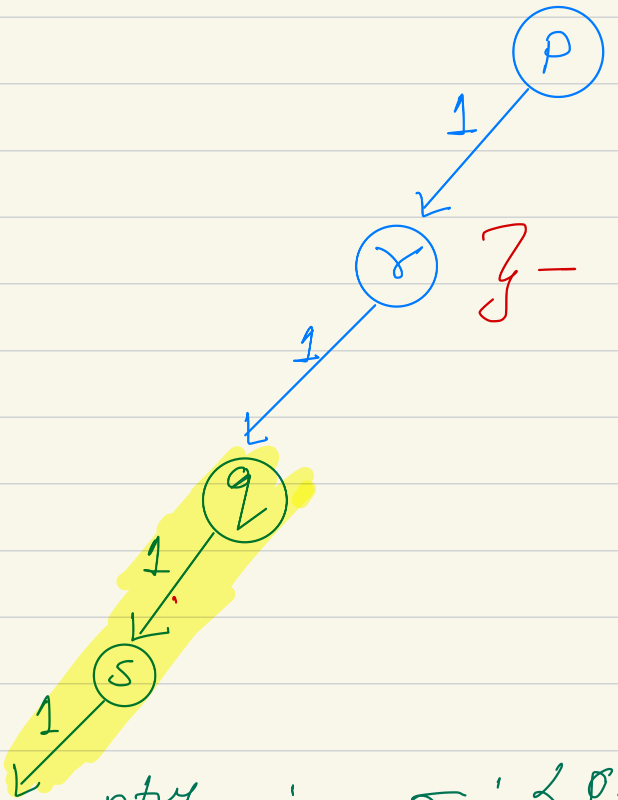
→ zChaff Solver ← efficient implementation of DPLL
2001, CAV paper
won test of time award.

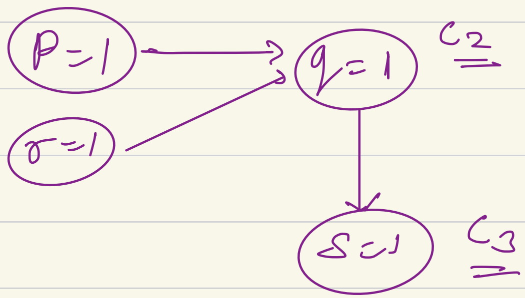
DPLL:

Complete & Backtracking-based algorithm.

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

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

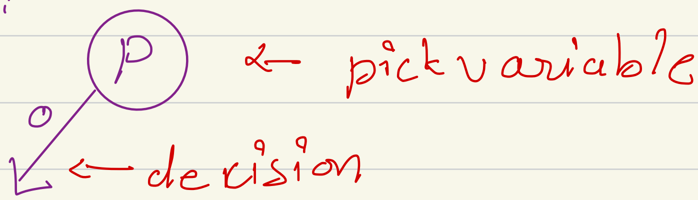




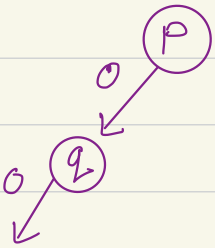
F is empty : $\sigma : \langle p=1, r=1, q=1, s=1 \rangle$, $\sigma \models F$

$F = (\neg p \vee q \vee r) \wedge (p \vee r \vee s) \wedge (p \vee r \vee \bar{s}) \wedge (p \vee \neg r \vee s)$
$\wedge (p \vee \neg r \vee \neg s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$

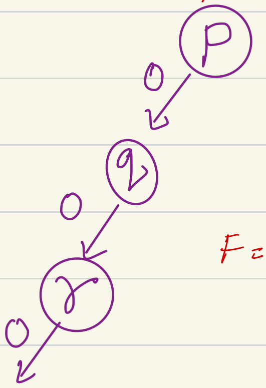
pick a variable, say p
& decide its polarity:



$F = (\neg p \vee q \vee r) \wedge (p \vee r \vee s) \wedge (p \vee r \vee \bar{s}) \wedge (p \vee \neg r \vee s)$
$\wedge (p \vee \neg r \vee \neg s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$

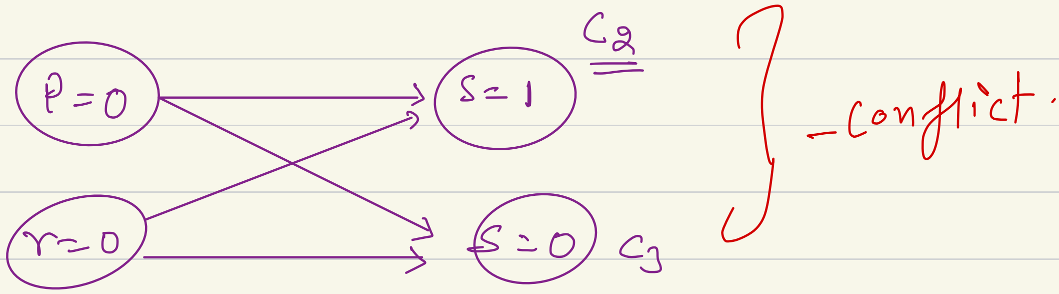


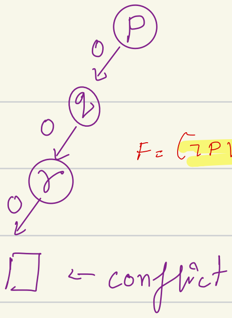
$F = (\neg p \vee q \vee r) \wedge (p \vee r \vee s) \wedge (p \vee r \vee \bar{s}) \wedge (p \vee \neg r \vee s)$
$\wedge (p \vee \neg r \vee \neg s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$



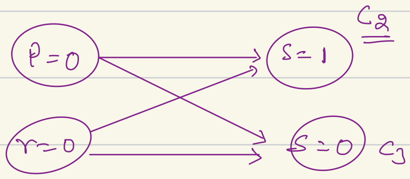
$F = (\neg p \vee q \vee r) \wedge (p \vee r \vee s) \wedge (p \vee r \vee \bar{s}) \wedge (p \vee \neg r \vee s)$
$\wedge (p \vee \neg r \vee \neg s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$

$\square$ ← conflict

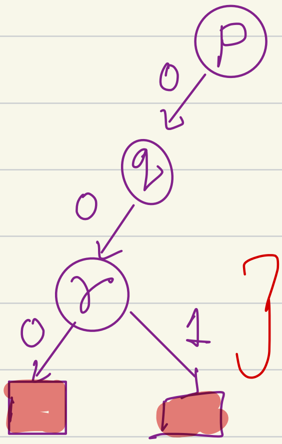




$F = (\neg p \vee q \vee r) \wedge (p \vee r \vee s) \wedge (p \vee r \vee \bar{s}) \wedge (p \vee \neg r \vee s)$
$\wedge (p \vee \neg r \vee \neg s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$

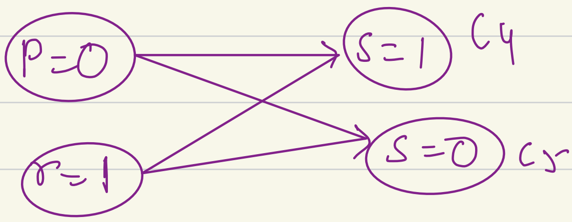


} conflict.

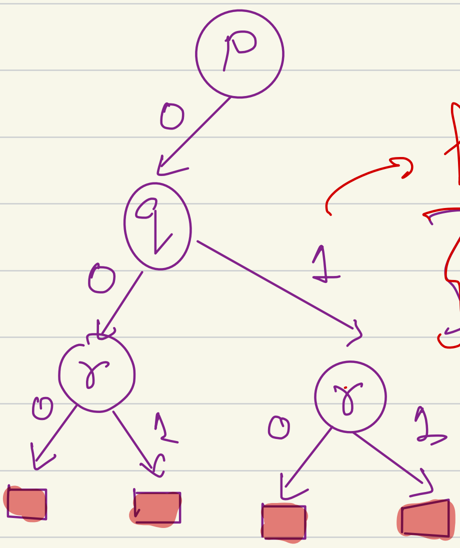


} backtrack to one level up & change the polarity.

$F = (\neg p \vee q \vee r) \wedge (p \vee r \vee s) \wedge (p \vee r \vee \bar{s}) \wedge (p \vee \neg r \vee s)$
$\wedge (p \vee \neg r \vee \neg s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$

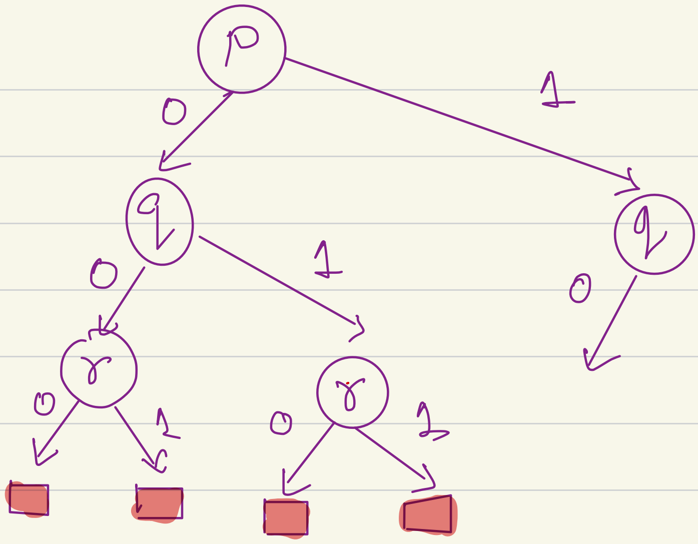


} conflict



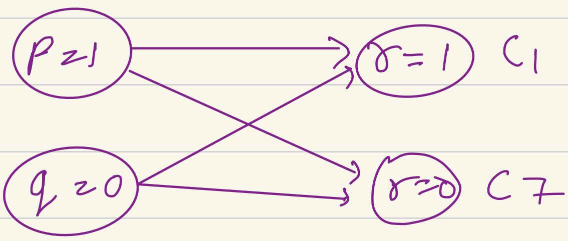
forced decision.

} backtracking to one level up & change the polarity

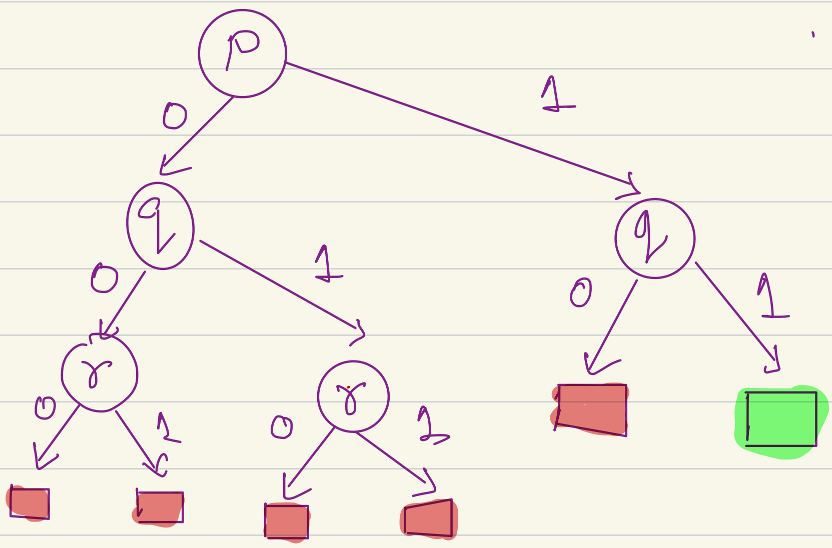


backtrack to one level up & changing the polarity

$$F = (\neg p \vee q \vee r) \wedge (p \vee r \vee s) \wedge (p \vee r \vee \bar{s}) \wedge (p \vee \neg r \vee s)$$
$$\wedge (p \vee \neg r \vee \neg s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$$

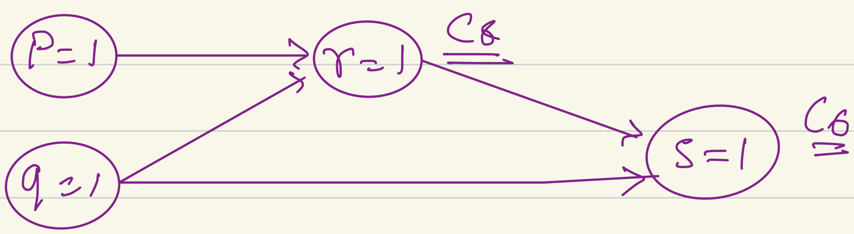


conflict.



backtracking.

$$F = (\neg p \vee q \vee r) \wedge (p \vee r \vee s) \wedge (p \vee r \vee \bar{s}) \wedge (p \vee \neg r \vee s)$$
$$\wedge (p \vee \neg r \vee \neg s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$$



DPLL

- → maintains a partial model m, initially ϕ
- → assigns unassigned variables 0 or 1 (randomly one after the other)
- → sometimes forced to choose assignments due to <u>unit literals</u>

```
DPLL (F, m)        // initially m is φ
{
   if F is true under m
          return SAT
   if F is false under m                 } backtracking at
          return UNSAT                       conflict

   if ∃ unit literal p under m then
          return DPLL (F, m [p → 1])       } unit
   if ∃ unit literal ¬p under m then         propogation
          return DPLL (F, m [p → 0])

   choose an unsigned variable p and a random
          bit b ∈ {0,1}                     }
     if DPLL (F, m [p ↦ b]) == SAT then      decision
          return SAT
   else
          return DPLL (F, m [p → 1−b])
}
```

Show a run of DPLL.

$F = (\neg P_1 \vee P_2) \wedge (\neg P_1 \vee P_3 \vee P_5) \wedge (\neg P_2 \vee P_4) \wedge$
$(\neg P_3 \vee \neg P_4) \wedge (P_1 \vee P_5 \vee \neg P_2) \wedge (P_2 \vee P_3)$
$\wedge (P_2 \vee \neg P_3 \vee P_7) \wedge (P_6 \vee \neg P_5)$

→ choose variable in order

$\langle P_6, P_7, P_1, P_5, P_2, P_4, P_3 \rangle$

choose polarity

$\langle P_6 = 0, P_7 = 0, P_1 = 1, P_5 = 1, P_2 = 0, P_4 = 1, P_3 = 1 \rangle$

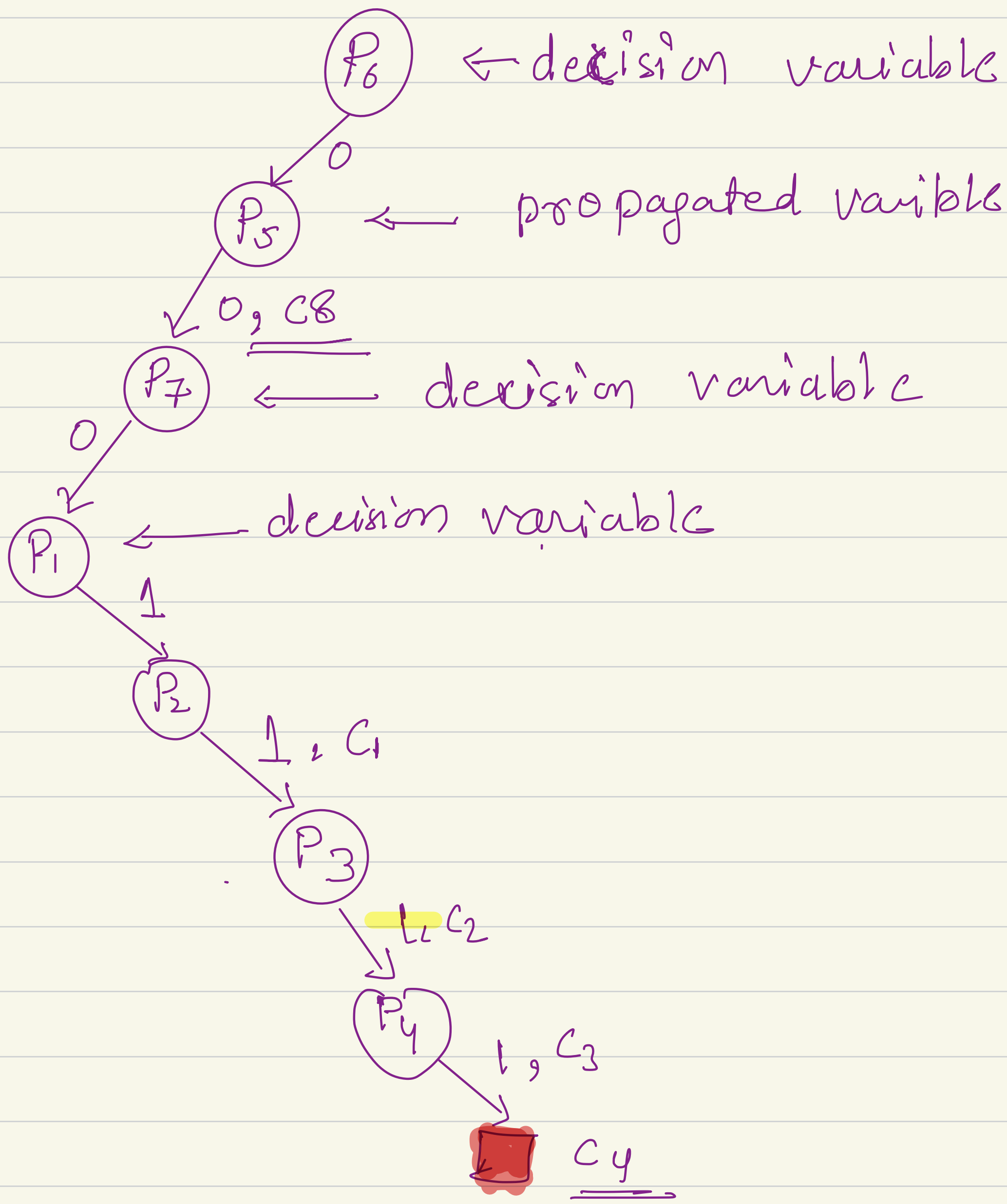
$$F = (\neg P_1 \vee P_2) \wedge (\neg P_1 \vee P_3 \vee P_5) \wedge (\neg P_2 \vee P_4) \wedge (\neg P_3 \vee \neg P_4) \wedge (P_1 \vee P_5 \vee \neg P_2) \wedge (P_2 \vee P_3) \wedge (P_2 \vee \neg P_3 \vee P_7) \wedge (P_6 \vee \neg P_5)$$

→ choose variable in order

$\langle P_6, P_7, P_1, P_5, P_2, P_4, P_3 \rangle$

choose polarity

$\langle P_6 = 0, P_7 = 0, P_1 = 1, P_5 = 1, P_2 = 0, P_4 = 1, P_3 = 1 \rangle$



DPLL Analysis

$n = |\text{variables of } F|$

1. Worst case time complexity:

$O(2^n)$

2. Best case time complexity:

$O(1)$

3. Worst case space complexity:

$O(n$

CDCL:

Conflict Driven Clause Learning

(Marques-Sliva and Karem A. Sakallah
1996-1999)

→ Build on top of DPLL

→ main difference is Backjumping is
non-chronlogical

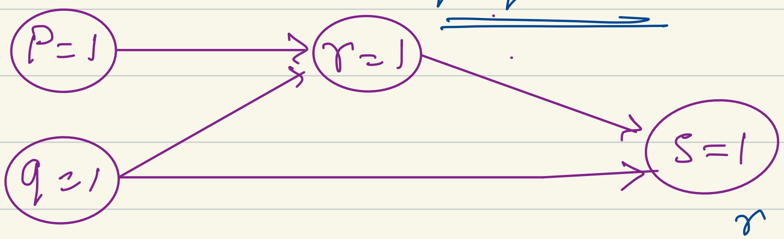
Implication Graph:

$G = (V, E)$

Each vertex in V represents a Boolean literal

Each directed edge from u to v represents if literal u is true, then the literal v is also true.

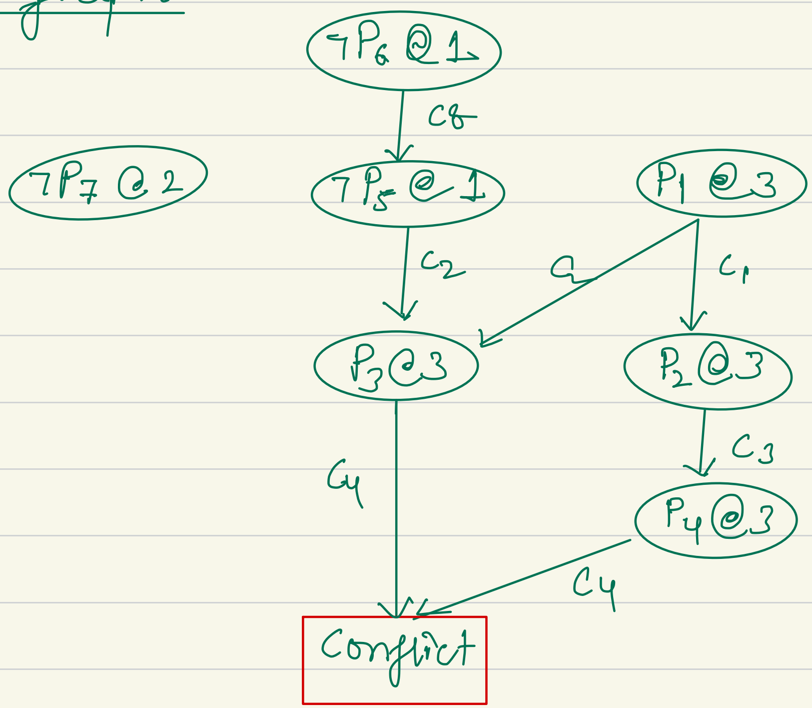
$p \wedge q \to r$



$r \wedge q \to s$

$$F = (\neg P_1 \vee P_2) \wedge (\neg P_1 \vee P_3 \vee P_5) \wedge (\neg P_2 \vee P_4) \wedge$$
$$(\neg P_3 \vee \neg P_4) \wedge (P_1 \vee P_5 \vee \neg P_2) \wedge (P_2 \vee P_3)$$
$$\wedge (P_2 \vee \neg P_3 \vee P_7) \wedge (P_6 \vee \neg P_5)$$

Implication graph



CDCL

1. Select a variable and assign randomly True or False
2. Do unit propagation
3. Build the implication graph
4. If there is a conflict
   - → Find the cut in the implication graph that led to the conflict.
   - → Derive a new clause → negation of assignments that led to conflict.
     conflict clause
   - → Non chronologically Backtrack to appropriate level
5. Otherwise continue with step 1, until all variables are assigned.