3. Mutation:

Flip the value.

→ Not done for every individual.

→ Incomplete -

→ Ant Colony Optimisation:

Read from book.

→ Particle Swarm optimization

→ Multi objective optimised problems - genetic algos are used.

↓

Pareto-optimal solutions.

S_1 - 80, 60 S_2 - 70, 70 S_3 - 50, 40

Taking S_1, S_3 → S_1 is better than S_3.

S_1, S_2 → S_1 is better on objective 1. S_2 is better on objective 2 } so, non dominating / pareto.

sessional-1

17/8/22 evening

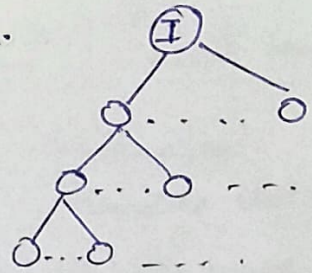
Constraint satisfaction problem (CSP):

1. Variables - $V_1, \ldots V_n$
2. Domains - $d_1, \ldots d_n$
   ↘ legal values of variables
3. Constraints on the variables.
   ↳ unary,
   binary,
   higher order (more than 2 variables).

Solution - An assignment of values to all the variables such that there are no constraint violations.

→ Graph coloring problem is CSP.

variables - vertices
Domains - {set of colours}
constraints - adjacent colours are diff.



BFS

search space

$n \times d$

$n(n-1)d^2$

$n(n-1)(n-2)d^3$

$\vdots$

$n! d^n$.

Normally, n nodes, each node having d possibilities

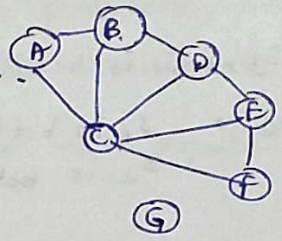
$\rightarrow d^n$

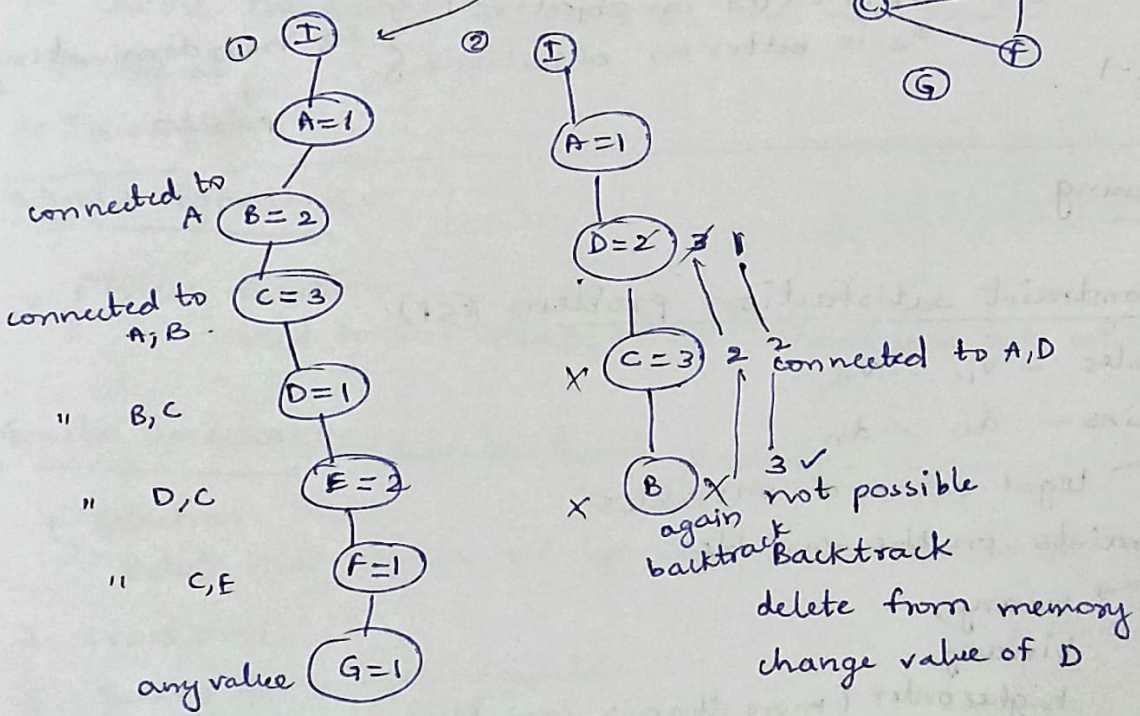
But above, $n! d^n >> d^n$.

Due to repetition of nodes.

→ DFS.

→ let colors = {1, 2, 3}, alphabetical order.





Backtracking search (DFS with slight modification).

→ Every CSP can be drawn into graph.

Constraint graph

1. Which variable should be considered next?
2. which value should be assigned to the chosen variable?

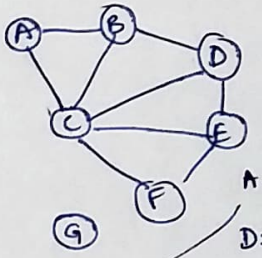
(i) choose variable with MRV
- minimum no. of remaining values.

If it contains 2 or more having same MRV
choose variable with high degree in the constraint graph
- Degree heuristic

If there is tie, choose any.

(ii) Least constraining value heuristic (LCV).

19/8/22



all are MRV.

	A	B	C	D	E	F	G
	123	123	123	123	123	123	123
A=1	1	23	23	123	123	123	123
D=2	1	3	3	2	13	123	123

made a wrong choice

Forward checking

	A	B	C	D	E	F	G
	123	123	123	123	123	123	123
C=1	23	23	(1)	23	23	23	123
B=2	3	(2)	(1)	3	23	23	123
D=3	3	(2)	(1)	(3)	2	23	123
E=2	3	(2)	(1)	(3)	(2)	3	123
A=3	(3)	(2)	(1)	(3)	(2)	3	123
F=3	(3)	(2)	(1)	(3)	(2)	(3)	123
G=1	(3)	(2)	(1)	(3)	(2)	(3)	(1)

any

Forward checking — Node consistency.

Complexity $\doteq (n-1)(d-1)d$ → constraint propagation level 1.

$= O(nd^2)$

	A	B	C	D	E	F	G
	123	123	123	123	123	123	123
A=1	1	23	23	123	123	123	123

LCV — D = ?

consider 1
affects B, C, E
possibilities = 2+2+2
= 6.

consider 2
possibilities = 1+1+2
= 4

consider 3
possibilities = 1+1+2
= 4

→ Arc Consistency: (Constraint Propagation level 2).

D = 2

D–E, D–B, D–C
2 1,3. 2 3 2 3

E affects F, C

E–C E–F

B–C
B–A

C–B
C–A
C–E
C–F

consider B–C
3 ~~3~~ becomes null → not consistent.

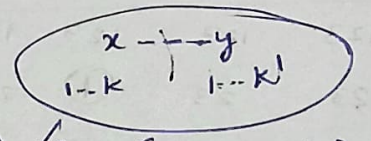
↓
Backtrack

(Refer book)

$O(n^2d^3)$

n^2 arcs,
1 arc can be inserted at most d times,
and d^2 comparisons

takes (d^2 comparisions)
d^2.



continue till either one of variable's domain becomes null, we backtrack.

else success if all arcs are consistent.

24/8/22 evening

$O(n^2 d^3)$

n^2 arcs and each arc can be checked atmost d times and d^2 comparisons b/w domains.

$x \cdots y$
$1..k \quad 1..k' \rightarrow d^2$ comparisons.

For ex: $z \rightarrow x$ checked.
$[z_1] \quad [x_1, x_2, x_3]$

now, $x \rightarrow y$
$[x_1, x_2, x_3] \quad [y_1, y_2]$

Now, here if x_1 doesn't give legal value. Remove x_1

x impacts z, so check again.

In this manner, each arc can be checked atmost d time

Chronological Backtracking

	A	B	C	D	E	F	G
	123	123	123	123	123	123	123
A=1	①	23	23	123	123	123	123
E=2	①	23	3	13	②	13	123.
NOW, E=1	①	23	23	23	①	23	123

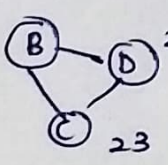
C has reduced domain.
D " " "
F " " "

$C - B^{23}$ ✓
$B - C^{3}$ ✓
$C - D^{13}$ ✓
$D - C^{3}$ ✓
$C - F^{13}$ ✓
$F - C^{3}$ ✓
$D - B^{2}$ ✓
$B - D^{1}$ ✓

A, E are already assigned.

(they use stack) for fewer checks.
can use queue for implementation

For, E=1,



→ Not colourable. (3 constitency).

→ Arc constitency can't determine all inconsistent nodes.

→ Can we backjump instead of backtracking?

Detection of early backtrack point → can be done by conflict set.
" " appropriate " "

Not taking structure info

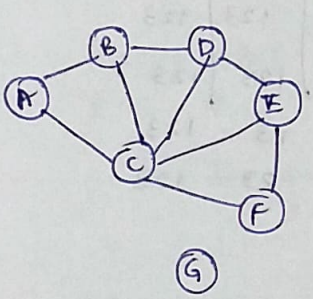
Conflict Directed Backjumping

→ Considering coloring tree/DAG is efficient

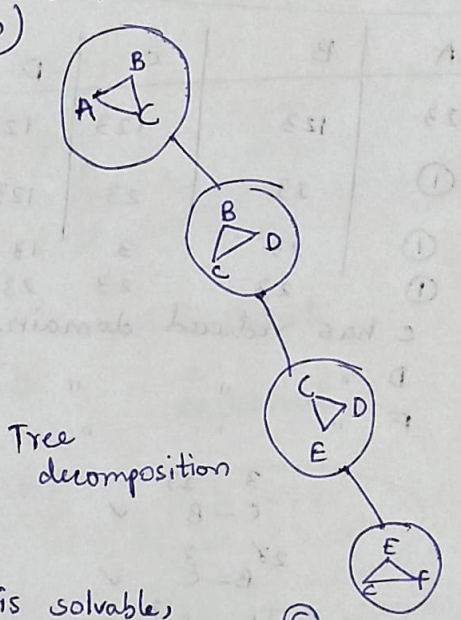
cut set = Remove few edges such that graph becomes tree.

→ Junction tree decomposition of graph – Refer book.

(Taking structure info)



G

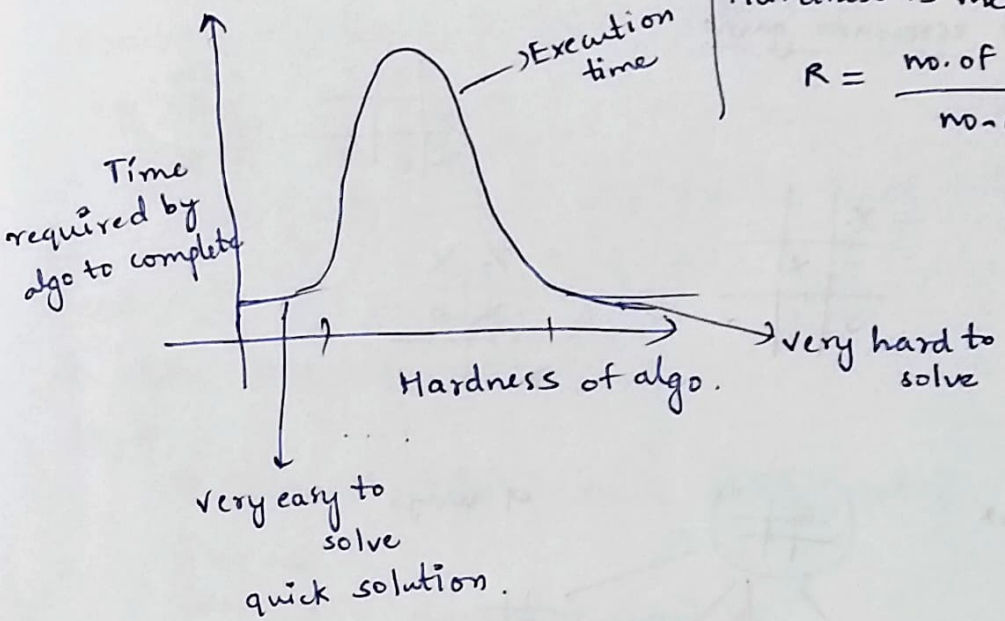


Tree decomposition

For 1st, check if subgraph is solvable,
goto 2nd, common variable's/node's values should be same.
and check if the subgraph is solvable.

G

25/8/22



Hardness is measured by ratio

$$R = \frac{\text{no. of constraints}}{\text{no. of variables.}}$$

Min-Conflicts Algo :- (Incomplete technique).

→ for any incomplete probs, first we have to see whether there is path requirement.
we should also have complete state formulation.

→ Use Local search

1. Randomly assign values to all the variables.
2. check if it's a solution.
3. If not,
   Find out all variables which are involved in a constraint violation.
   Randomly pick one variable
   Assign it the minimum conflicting value
   check if it is a solution
   If not,
   Repeat.

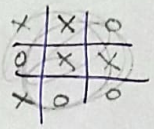
→ We can do forward checking for choosing min. conflicting value

→ static search formulation

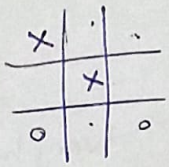
→ What do you mean by state
action
goal state
successor action

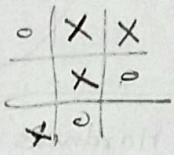
Game playing:

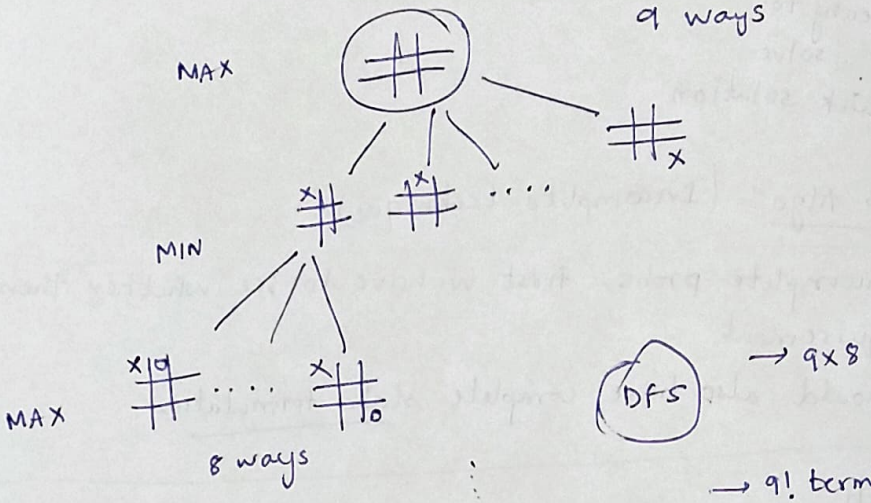
→ Two player zero-sum game:



strategy





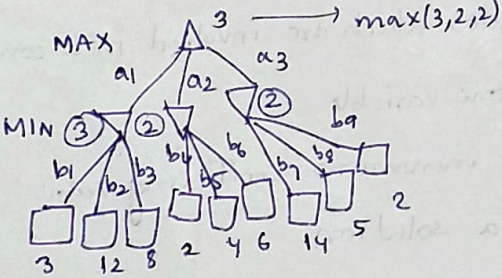


Mini-max search algo.

△ – MAX

▽ – MIN

26/8/22



MIN tries to minimise

If MAX choses a_1, then min has b_1, b_2, b_3.

take least.

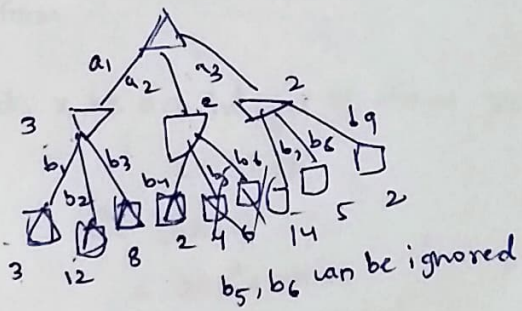
same goes to a_2, a_3.

Mini-max algo:-

```
function minimax-decision (state)
begin
    return the action a in
    Actions(state) that maximises MIN_VALUE(Result(a, state)),
end
function MAX_VALUE(state)
begin
    if Terminal(state) return Utility(state),
    V = -∞
    for a,s in successors(state) do
        V = Max(V, MIN_value(s))
    return v;
end

function MIN_VALUE(state)
begin
    if Terminal(state) return Utility(state)
    V = ∞
    for a,s in Successors (state) do
        V = Min( V, MAX_VALUE(s)) ;
    return v;
end.
```

→ It explores every branch in DFS manner till terminal state.



b_5, b_6 can be ignored

α-cutoff - for MAX

β-cutoff - for MIN.

Minimax with α,β cutoff - refer book.

pruning

→ Properties of mini-max algo:

→ complete

→ optimal (assuming second player is playing optimal)

→ size of game tree is exponential.

Polynomial space → $O(bm)$

Exponential time → $O(b^m)$

b - branching factor

m - max depth.

13/9/22

Agent design in semi-observable env:

→ **Propositional logic:**

symbols which are propositions which take either T or F.

→ **logical connectives**

To connect symbols

and ≡ $A \wedge B$

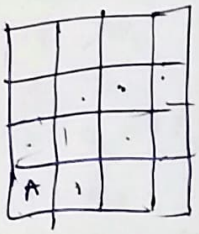
or ≡ $\vee$

NOT ≡ $\neg$

implies ≡ $\rightarrow$

iff ≡ $\leftrightarrow$

(if only if)

→ 

If agent A is alive in (1,1),

P – for pit presence

W – for wumpus presence

S – adjacent of wumpus

B – adjacent of pit.

$\neg P_{11}$ (no pit)

$\neg W_{11}$ (no wumpus)

$\neg B_{11}$ (no breeze) ⟶ Background knowledge.

$\neg S_{11}$ (no stinge)

$\neg G_{11}$ (no glitter)

$P_{11} \rightarrow B_{12} \wedge B_{21}$ (not iff bcoz $B_{12} \wedge B_{21} \rightarrow P_{22}$ also)

$B_{11} \leftrightarrow P_{12} \vee P_{21}$

→ **Entailment:**

Reasoning on background knowledge. ⟶

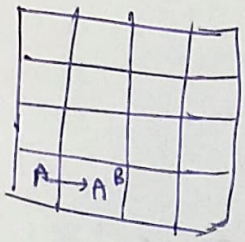
→ Knowledge base → Background knowledge → percepts.

$KB \overset{?}{\models} \neg P_{12}$

KB ↙ Knowledge Base

A	B	A→B	A↔B
T	T	T	T
T	F	F	F
F	F	T	T
F	T	T	F

$x \leftrightarrow y \equiv (x \rightarrow y) \wedge (y \rightarrow x)$



$B_{12}, Right_{11}$

$B_{12} \leftrightarrow P_{11} \vee P_{13} \vee P_{22}$

$KB \overset{?}{\models} \neg P_{13}$

$KB \overset{?}{\models} \neg P_{22}$

∴

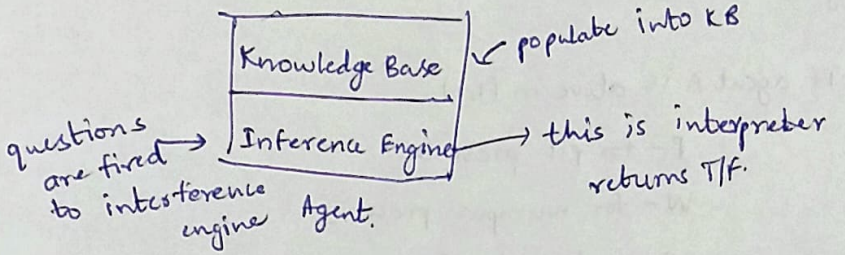
Include the taken action in KB (Knowledge Base).

Populate KB.

→ Ask right questions, action, populate until goal.

→ Declarative approach to programming.

→ How entailment works:



→ Logician's view of entailment:

Models: model represents the world at a given instant.

ex: $\neg P_{11} + \neg B_{11}$ + Background knowledge.

$B_1 \leftrightarrow P_{12} \vee P_{21}$. KB

$\neg P_{12}$ — 'α' (say)

$KB \models \alpha$ iff

Models (KB) ⊂ Models(α)

subset of

⊨ - entailment

	P_{11}	B_{11}	P_{12}	P_{21}	$B_{11} \leftrightarrow P_{12} \vee P_{21}$	KB	$\alpha = \neg P_{12}$
1.	F	F	T	T	F	F	F
2.	F	F	T	F	F	F	F
3.	F	F	F	T	F	F	T
4.	F	F	F	F	T	T	T

KB → $\neg P_{11} + \neg B_{11}$
For KB to be true,
P_{11}, B_{11} – F.

Models(KB) = {4}

Models(α) = {3,4}

Models(KB) ⊂ Models(α).

14/9/22

KB

1. $\neg P_{11}$
2. $\neg B_{11}$
3. $B_{11} \leftrightarrow P_{12} \vee P_{21}$
4. B_{12}
5. $B_{12} \leftrightarrow P_{11} \vee P_{13} \vee P_{22}$

$\alpha = \neg P_{13}$

$\neg B_{11}$ – T
B_{11} – F

$KB \stackrel{?}{\models} \alpha$

Interference By Enumeration

P_{11}	B_{11}	B_{12}	P_{12}	P_{21}	P_{13}	P_{22}	$B_{12} \leftrightarrow P_{11} \vee P_{13} \vee P_{22}$	KB	$\alpha = \neg P_{13}$
F	F	T	F	F	T	T	T	T	F
f	F	T	F	F	T	F	T	T	F
F	F	T	F	F	F	T	T	T	T
F	F	T	F	F	F	F	F	F	T

→ sound and complete for checking entailment.
✓ doesn't provide false conclusions

→ Time complexity = $O(2^n)$

→ space – linear

→ DFS manner

→ **Theorem proving**

→ 2nd method for entailment

We have to prove α, given KB.

Axioms:

1. $T \vee x = T$
2. $f \vee x \equiv F$
3. $T \wedge x \equiv x$
4. $F \wedge x \equiv F$
5. $x \wedge y \equiv y \wedge x$
6. $x \vee y \equiv y \vee x$
7. $((x \wedge y) \wedge z) \equiv (x \wedge (y \wedge z))$
8. $((x \vee y) \vee z) \equiv (x \vee (y \vee z))$
9. $\neg(\neg x) \equiv x$ (negation elimination)
10. $x \rightarrow y \equiv \neg x \vee y$ (implication elimination)
11. $x \rightarrow y \equiv \neg y \rightarrow \neg x$ (contraposition)
12. $x \leftrightarrow y \equiv (x \rightarrow y) \wedge (y \rightarrow x)$ (double implication elimination)
13. $\neg(x \vee y) \equiv \neg x \wedge \neg y$ (Demorgan's law)
14. $\neg(x \wedge y) \equiv \neg x \vee \neg y$ (demorgan's law)
15. $(x \wedge (y \vee z)) \equiv (x \wedge y) \vee (x \wedge z)$ - Distributivity of $\wedge$ over $\vee$.
16. $(x \vee (y \wedge z)) \equiv (x \vee y) \wedge (x \vee z)$ - Distributivity of $\vee$ over $\wedge$.

---

17. $(x \rightarrow y) \wedge x \equiv y$ (Modus Ponens)
18. written as $\dfrac{x \rightarrow y,\ x}{y}$.
19. $x \wedge y \equiv x$ and $x \wedge y \equiv y$ } AND elimination.

written as $\dfrac{x \wedge y}{x}$ and $\dfrac{x \wedge y}{y}$.

} Reasoning patterns.

→ write truth tables for checking soundness of axioms.

Theorem proving ex:

KB

1. $\neg P_{11}$
2. $\neg B_{11}$
3. $B_{11} \Leftrightarrow P_{12} \vee P_{21}$

$\alpha \equiv \neg P_{12}$

By above axioms, we can modify 3.

By axiom 12. $(B_{11} \rightarrow P_{12} \vee P_{21}) \wedge ((P_{12} \vee P_{21}) \rightarrow B_{11})$

By axiom 10 $\rightarrow (\neg B_{11} \vee P_{12} \vee P_{21}) \wedge ((P_{12} \vee P_{21}) \rightarrow B_{11})$

$\Rightarrow (\neg B_{11} \vee P_{12} \vee P_{21}) \rightarrow$ (6) (4) (5)

$(\neg(P_{12} \vee P_{21}) \vee B_{11}) \rightarrow$ (7)

8. Demorgan's on 7 $\Rightarrow \neg P_{12} \wedge \neg P_{21} \vee B_{11}$
9. as we know $\neg B_{11} \equiv T$

   so, from 6, $T \vee P_{12} \vee P_{21} \equiv T$
10. Use contraposition (axiom 11) on (5)

    $\neg B_{11} \rightarrow \neg(P_{12} \vee P_{21})$
11. Use modus ponens on (3) and (10)

    $\neg(P_{12} \vee P_{21})$
12. Demorgan's on (11) $\neg P_{12} \wedge \neg P_{21}$
13. Use and-elimination, $\neg P_{12} \equiv \alpha$

$\neg P_{12} \wedge \neg P_{21} \equiv \neg P_{12}$

$\neg P_{12} \wedge \neg P_{21} \equiv \neg P_{21}$

$KB \vdash \alpha$

$KB \vdash \alpha$ — inference by enumeration

$KB \vdash \alpha$ — Theorem proving

→ Both are sound and complete.

→ Theorem proving is sound bcoz axioms are themselves sound

→ is complete when all 18 axioms are included.

→ Validity of a formula in propositional logic:

Let s be sentence
If for every T|F inputs, output is true
then s is true.

→ Deduction Theorem:

KB ⊨ α iff KB → α is valid

→ Satisfiability of a formula in PL:

Draw truth table for all propositions of s,
if one of the outputs is true → satisfiability.

KB ⊨ α iff KB ∧ ¬α is unsatisfied.

(negation of deduction theorem)

↓

corollary of deduction.

16/9/22

Inference Techniques in PL:

1. Truth table enumeration.
2. Theorem proving.

Deduction Theorem:

KB ⊨ α iff KB → α is valid
KB ⊨ α iff KB ∧ ¬α is unsatisfiable

→ Conjunctive Normal Form (CNF):

A set of clauses connected using '∧'.

clause:

1. A single literal (either +ve/-ve)
   ex: x, ¬y
2. Multiple literals (either +ve/-ve)
   connected using 'v'.

ex: CNF → $(\neg x) \wedge (y \vee z \vee \neg w) \wedge (\neg y \vee x \vee z)$

Ex:

1. $\neg P_{11}$ 2. $\neg B_{11}$ 3. $B_{11} \leftrightarrow P_{12} \vee P_{21}$

facts already in clause

4. $\alpha : \neg P_{12}$.

For 3rd,

(i) Eliminate bimplication

$$(B_{11} \rightarrow (P_{12} \vee P_{21})) \wedge ((P_{12} \vee P_{21}) \rightarrow B_{11})$$

(ii) Eliminate $\rightarrow$

$$(\neg B_{11} \vee (P_{12} \vee P_{21})) \wedge (\neg (P_{12} \vee P_{21}) \vee B_{11})$$

Removal of bracket

$$(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg (P_{12} \vee P_{21}) \vee B_{11})$$

↓ clause

(iii) Demorgan's law.

$$(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge ((\neg P_{12} \wedge \neg P_{21}) \vee B_{11})$$

(iv) Distributivity

$$(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg P_{12} \vee B_{11}) \wedge (\neg P_{21} \vee B_{11}) \rightarrow \text{(CNF)}$$

3a 3b 3c (say)

→ Resolution Rule:-

Resolve 2 clauses at a time and one term at a time.

$$\frac{x_1 \vee x_2 \vee \ldots \vee x_i \vee \ldots \vee x_m \;(c_1), \quad y_1 \vee y_2 \vee \ldots \vee y_j \ldots \vee y_n \;(c_2)}{c_3 - \; x_1 \vee \ldots \vee x_{i-1} \vee x_{i+1} \ldots \vee x_m \vee y_1 \vee y_2 \ldots \vee y_{j-1} \vee y_{j+1} \ldots \vee y_n}$$

if $x_i = \neg y_j$.

ex: (c_1, c_2)

1. $a \vee \neg b \vee \neg c$, $c \vee p \Rightarrow$ $c_3 - a \vee \neg b \vee p$
2. $a \vee \neg b$, $\neg a \Rightarrow \neg b$.
3. $a, \neg a \Rightarrow \square$ (null clause) – False

→ Resolution is sound.

→ From above example,

$KB \models \alpha$ iff $KB \wedge \neg\alpha$ is unsatisfiable

RC(s) – Resolution closure for set of clauses.

1. Keep on resolving 2 clauses at a time
2. Get clauses until it can't be resolved.
3. That is RC

→ It is finite set.

From above ex:

1. $\neg P_{11}$ 2. $\neg B_{11}$ 3. $B_{11} \leftrightarrow P_{12} \vee P_{21}$ 4. $\alpha : \neg P_{12}$

$\neg\alpha : P_{12}$

To prove: $KB \wedge \neg\alpha$ is unsatisfiable $\leftrightarrow KB \models \alpha$

3. $(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg P_{12} \vee B_{11}) \wedge (\neg P_{21} \vee B_{11})$

3a 3b 3c

3b 2 Resolve

$\neg P_{12}$ 4

□ (null clause)

∴ We just need one single rule.
No need of all rules.

→ NP complete

Ex: 1. $\neg P_{11}$ 2. $\neg B_{11}$ 3. B_{12} 4. $B_{12} \leftrightarrow P_{11} \vee P_{13} \vee P_{22}$

5. $\neg\alpha \equiv P_{13}$.

convert to CNF

$4 \Rightarrow \quad B_{12} \leftrightarrow P_{11} \vee P_{13} \vee P_{22}$

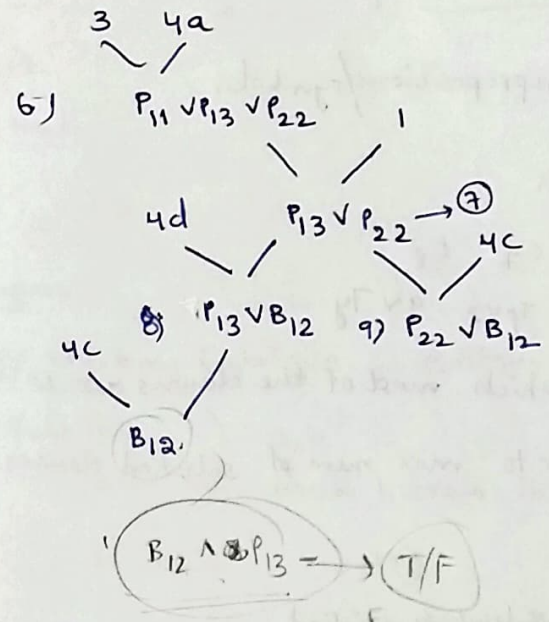
$(B_{12} \rightarrow P_{11} \vee P_{13} \vee P_{22}) \wedge ((P_{11} \vee P_{13} \vee P_{22}) \rightarrow B_{12})$

$(\neg B_{12} \vee (P_{11} \vee P_{13} \vee P_{22})) \wedge (\neg(P_{11} \vee P_{13} \vee P_{22}) \vee B_{12})$

$(\neg B_{12} \vee P_{11} \vee P_{13} \vee P_{22}) \wedge ((\neg P_{11} \wedge \neg P_{13} \wedge \neg P_{22}) \vee B_{12})$

$(\neg B_{12} \vee P_{11} \vee P_{13} \vee P_{22}) \wedge ((\neg P_{11} \vee B_{12}) \wedge (\neg P_{13} \vee B_{12}) \wedge (\neg P_{22} \vee B_{12})$

4a 4b 4c 4d



$\neg(P_{13} \wedge \neg B_{12}) \rightarrow T$

F F

Refutation completeness of Resolution for PL

$B_{12} \wedge P_{13} \rightarrow T/F$

F

→ We didn't get null clause.

$KB \vdash \alpha$ is not entiled.

21/9/22

Refer priya notes

22/9/22

Local search for Boolean satisfiability:

→ Let, $c_1 \wedge c_2 \wedge \ldots \wedge c_n$, m prepositions/symbols.

Taking random clauses

for ex, c_3 c_7 c_8

$\neg x \vee y \vee \neg z$ $\neg p \vee z$ $q \vee \neg y$

Hill climbing:

changing the literal for which most of the clauses are satisfied.

Pick the literal which leads to max. num of satisfied clauses

Walk-Sat:

~~Random clauses~~

check clauses which are false/not satisfied

Take literals

Randomly pick one clause out of not satisfied clauses.

Take literal with prob. p, flip its value.

Otherwise, pick the symbol that leads to max. no. of satisfied clauses

Iterate

→ If we don't get all satisfied clauses, we can't conclude that model doesn't exist. we can just tell it is not complete

→ **Horn clauses:-**

If all clauses are in same form, then boolean satisfiability can be solved in polynomial time.

→ Horn clause is one in which there is atmost one +ve symbol.

Ex: x, $\neg x$, $\neg x \vee \neg z$

$\neg x \vee \neg z \equiv \neg x \vee \neg z \vee F$

$\neg(x \wedge z) \vee F$

$(x \wedge z) \rightarrow$ False

Ex: (KB)

1. $p \rightarrow q$
2. $l \wedge m \rightarrow p$
3. $b \wedge l \rightarrow m$
4. $a \wedge p \rightarrow l$
5. $a \wedge b \rightarrow l$
6. a
7. b

Find q

Antecedent → consequent

Techniques

1. Forward checking technique — Bottom up approach.

Take list, L: a,b

scan KB., check literals inferred to be true given a,b are true

L: a,b,l (5th)

scan KB again, given a,b,l are true

L: a,b,l,m (3rd)

L: a,b,l,m,p (2nd)

L: a,b,l,m,p,q (1st)

Problems:

If KB is huge, we keep on adding more symbols to list, which may not require to prove the literal to be true.

→ Better to ask what are required to prove literal is true

2. Backward checking:

L: q (To prove q, we need p)

L: P (we need l,m)

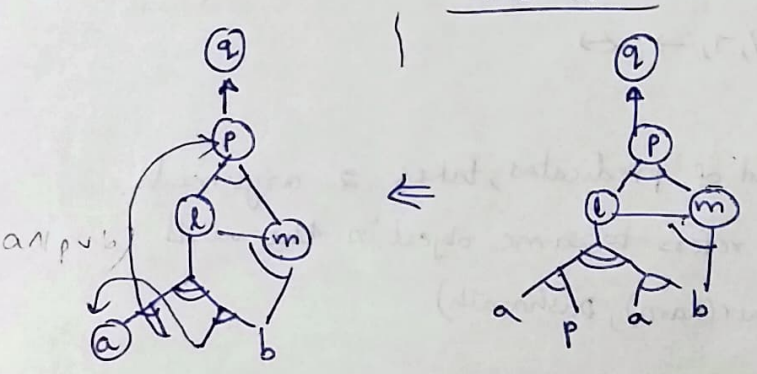
L: l,m

a,p — b,l (repeated, not required)

OR

a,b

AND-OR Tree



~ ≡ AND

≳ ≡ OR

Sessional-2