Local Search Techniques - only for problems that are computationally hard.

Page No.: Date: YOUVA

Particle Swarm Optimisation (PSO)

Ant Colony Optimisation (used in routing etc)

Optimal path has a large conc. of feromone
↓ from food source to ant hill ↓ evaporates with time.

Encode solution as an ant.

Bird Flocking

Local leaders followed by flocks
Global leader - circulating

L-13
21/08/19

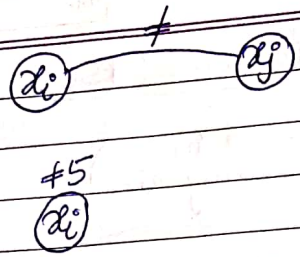
Constraint Satisfaction Problem (CSP)

set of variables $\equiv X\{x_1, x_2 \ldots x_n\}$.
domains of variables $\equiv D\{d_1, d_2 \ldots d_n\}$.

constraints on variables
- Unary constraint - $v(x_5) \neq 5$ or x_4 is even.
- Binary constraint - $v(x_5) \neq v(x_6)$
- Higher order constraint - $x_4 \neq x_5 \neq x_6$

CSP - legal values should be assigned to all variables such that there are no constraint violations

We can draw a constraint graph for a CSP.



$x_i \neq x_j$ is the constraint

$x_i \neq 5$
↓
constraint

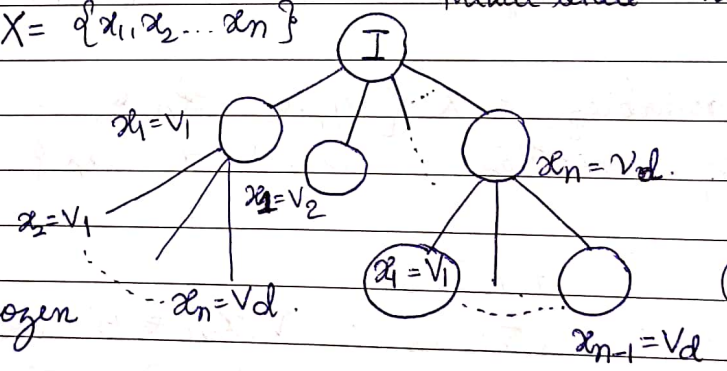
Hypergraph - if more than one constraint b/w two variables.

Given the constraints, variables and domains, we need to find an assignment of values to satisfy all constraints

We can model CSP as search problem.

Assume that domain is finite, discrete and size is 'd'.

$X = \{x_1, x_2 \dots x_n\}$



goal node at depth n.

n & d^n.

Goal - All nodes at depth n, we don't need to go beyond that depth. - finite depth.

∴ We can use DFS here.

Backtracking search

1. Choose one of the variables x_i
2. Choose one of the possible values for x_i such that no constraints are violated.

If we exhaust all possible values of x_i (for every value there is a violation of constraint), we cannot choose some other variable to assign since anyway all variables need to be assigned a value in CSP.
∴ We backtrack and choose a different value for x_{i-1} and try again.
check for violations → again start DFS.

There might be optimization in ~~CF~~ CSP such that some values are preferred for certain variables.

If there is a failure, we go back and forget about value assigned - store only those values/assignments that do not violate any constraints
⇒ DFS (only one branch active at a point).

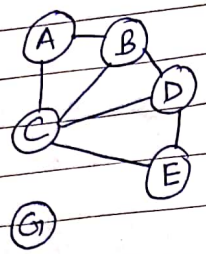
Which variable to assign first - how to make an informed choice?

⇒ Variables with least no. of possible values is most constrained and need to be assigned first.

1. Minimum number of remaining values (MRV) heuristic
Choose MRV variable.

Eg) Graph colouring - MRV - same no. of remaining values (colours) for all nodes in the beginning.

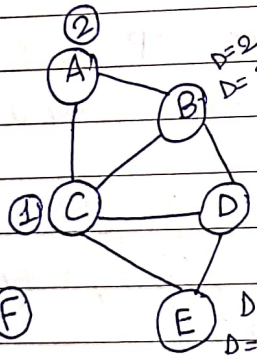
Can we choose randomly ?



⇒ C is assigned first
⇒ It will affect max^m no. of vars' remaining values
⇒ most constrained
⇒ highest degree.

If we have multiple MRV variables, choose one with the highest degree (as per the constraint graph). In case of a tie, choose one randomly.

If we now have MRV – which value to choose for it?
↓
see the impact of this value on other variables.
↓
choose value with least impact



D=2 B={3}.
D=3 B={2}.
possibility {2,3}.

D=2 {3}=B
D=3 B={2}.

⇒ For D=2, both B and E have options. (Min^m colours).
⇒ For D=3, B will not have option but to go for fourth colour.

∴ We choose D=2.

2) Least Constraining Value (LCV) heuristic
Assign that value which has the highest number of remaining values for the variables that have not yet been assigned.

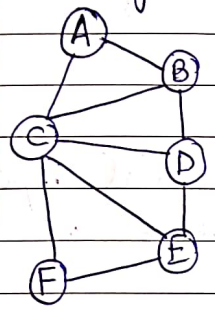
L-14
22/08/19.

CSP

1) MRV heuristic
In case of tie of MRV variables, choose variable with highest degree in constraint graph.

2) LCV heuristic
choose that value from possible values that leads to largest number of choices for variables that have not yet been assigned

3 colourability:



	(1,2,3) A	1,2,3 B	1,2,3 C	1,2,3 D	1,2,3 E	1,2,3 F
C=1	2,3	2,3	(1)	2,3	2,3	2,3
B=2	3	(2)	(1)	3	2,3	2,3
D=3	3	(2)	(1)	(3)	2	2,3
E=2	3	(2)	(1)	(3)	(2)	3
A=3	(3)	(2)	(1)	(3)	(2)	(3)
F=3						

discrete finite CSP – enumerate all values.

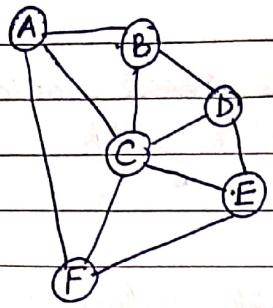
forward checking – omitting values that no longer are available to check.
In the first step, we can choose 1/2/3 – scenario remains same.

n: nodes
d: colours.

This table: size
number of columns : $n * d$
number of rows : n
size = $n^2 d$ (polynomial).

Choosing MRV : $(n-1)d$.
Choosing LCV : $(n-1)d * d$.

→ updation of table can be done in polynomial time.



	A	B	C	D	E	F
	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3
C=1	2,3	2,3	①	2,3	2,3	~~1~~,2,3
~~A=2~~	②	~~3~~	①	~~2,3~~	2,3	~~1~~,2
~~B=3~~	②	③	①	2	2,3	~~3~~
~~D=2~~	②	③	①	②	3	3
~~E=3~~	②	③	①	②	③	—
A=3	③	2	①	2,3	2,3	2
~~B=2~~	③	②	①	3	2,3	2
~~D=3~~	③	②	①	③	~~2~~	2
~~E=2~~	③	②	①	③	②	—

failure. ∴ we ← backtrack.

↓

keep backtracking till we get some other possible value.

C=1
A=2
constraint prop^n.

This is the reason why this works in polynomial space because backtracking ∄ the assigned values get deleted.

After second assignment also fails, there are no other options available. ∴ No solution

Constraint propagation → removing value from possible values of variables due to some assignment.

Can we do further propagation?

We check for variables whose domain get reduced if they are connected in the constraint graph.

	A	B	C	D	E	F
	(1,2,3)	(1,2,3)	(1,2,3)	(1,2,3)	(1,2,3)	(1,2,3).
C=1	2,3	2,3	①	2,3	2,3	2,3
A=2	②	3	①	2,3	2,3	3
constraint prop^n.	②	③	①	2,3	2,3	—

if B and F were connected (in constraint graph) (B and F are reduced here), we check whether both can be assigned legal values.

Arc consistency

check for every value of X_i – is there reduction in legal values of X_j? If yes, and reduction is such that domain of X_j becomes null, there is no arc consistency, remove that value of X_i from domain
∴ we backtrack.

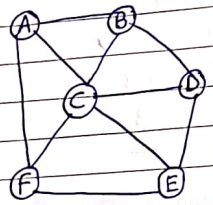
X_i —— X_j
[1,2....k] [1,2...k'].

Some X_k might be connected to X_j and its values might get affected due to reduction in domain of X_j – we keep propagating this constraint till we get fixed point.
↳ no changes in domains.

if X_j is selected next.

L-15
26/08/19

which all vertices a node that has had reduction connected to



3 colours available.
constraint propogation

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

When A, B, D, E, F have been reduced, check all arcs (bidirectional) to see if constraint needs to be propogated.

A=2 ② 3 ① 2 3 2, 3 3.
↓ reduced ↓ reduced.

Check for B → all vertices connected to B

B → D
[3] → [2,3]. No problem

D → B (bidirectional)
[2,3] [3]

∴ D=2 has to be done. Constraint is propogated.

② 3
② 3
E → F
[3] [3]
E=3. cann

n : no. of
d : degree

O(n²) arcs.

every arc is t
d times
For every ch
possibilities.
for every
need to c

∴ O(n

Optimisation i
Backjumping
variable w

ices a node that has
connected to

colours available.
onstraint propogation

F.
123
23.

reduced, check
o see if constraint

2,3 3.
↓
reduced.

connected to B

Constraint is

② 3 ① 2 2,3 3 → domain of E reduces when D=2.
② 3 ① 2 3 3

E → F
[3] [3].
E=3. cannot be given.

E domain becomes null, we backtrack.

. n : no. of variables.
d : ~~degree~~ / domain size.

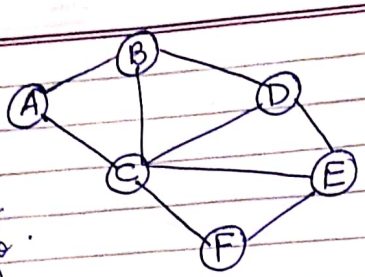
$O(n^2)$ arcs.

every arc is to be checked 'd' times
For every check – d possibilities.
for every possibility, we need to check for legal value – 'd' ~~leg~~ values checked

$\therefore O(n^2 \cdot d^3)$

$X_i \rightarrow X_j$
check for every value of X_i whether there is a legal value of X_j.
$O(d^2)$ operations.

Optimisation in the way backtracking is done.
Backjumping – jump multiple steps earlier to find variable whose value needs to be changed.



just to see arc consistency, not acc to algo.

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

3.

B domain reduced. becomes null.

E = 1 → arc consistency check is positive even though it is not a correct assignment.

∴ Arc consistency is not always efficient

At every assignment - we keep a list of variables that are impacted → conflict set.

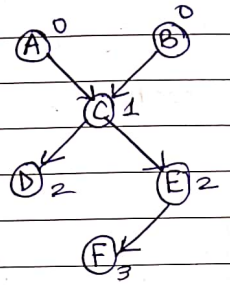
X → conflict set of X.
Y
Z
P
Q
R

After assigning R, M becomes null. (constraint propogation) (domain)

We go back in the history, look at conflict set of R. If R ~~has~~ does not have M in its conflict set,

we don't need to backtrack to R.
Check for Q – and so on.
Assignment history – latest variable with M Rn conflict set – unlds all assignments till we get there. i.e backjump to that variable.

This approach is called conflict directed backjumping



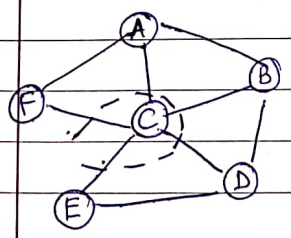
Multi tree
Unrooted tree

if parents have different levels
1 + max {parents levels}.

for odd levels – 1 colour
for even levels – 1 colour

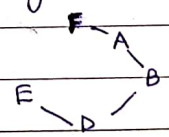
No cycles here.

Try to bring graph to this form by removing certain edges – cut set.
Give assignments – try to include cut set. after this



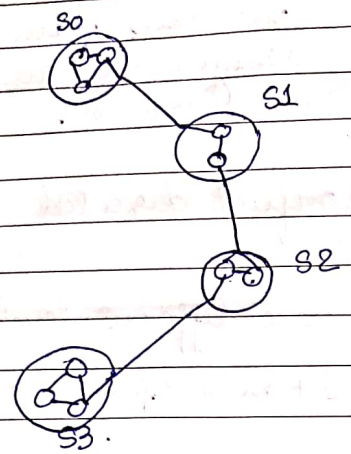
cut set = {C}. → cutset should be of minimum size.

Give assignment to the multitree and try to colour cutset while accomodating this assignment.



Page No.:
Date:

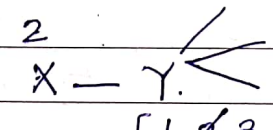
Tree Decomposition of a Graph



These are supernodes.
If supernodes form a tree, we try to find solution to subproblems
↓
inside supernode.
We will then stitch the solution together.

All the algorithms discussed so far are complete.
Can we use local search techniques for CSP?

Any sol^n of CSP can be made a vector.
We randomly initialise this vector.
In case there are constraint violations, we choose the variable which is in most number of constraint violations i.e max^m arcs that are inconsistent



choose {1, 3, 4} one from this set which is giving least number of violations.

The state changes.
Then we iterate — Min Conflicts approach.
↓
incomplete.
does not guarantee a solution

Stochastic approach of CSP.

Time req'd to reach a sol^n.

TSP

Scanned by CamScanner

er nodes.
: form a tree,
find solution
lems.
upernode.
stitch the
ther.

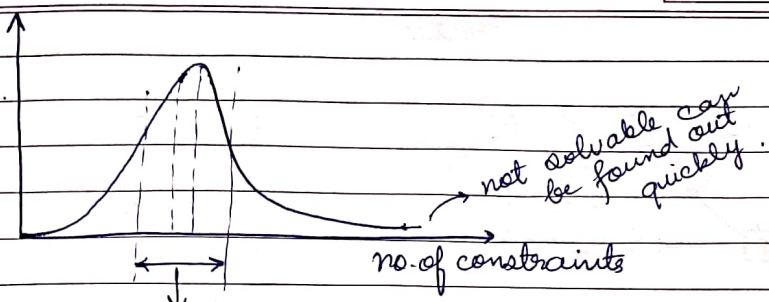
re complete.
er CSP ?

er

, we choose
nber of
es that are
one from
ich is
t number
ns.

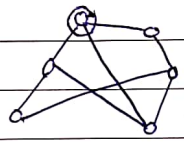
approach.

te.
uarantee a
tion



Give local search technique - a toy. If we get a solution, well and good - else go for complete algorithm.

TSP



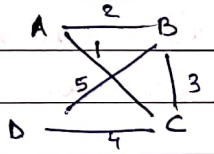
no. of cities * shortest distance.
unvisited

admissible - cannot overestimate.
polynomial time

NO.
MST use karna hoga

Matrix - path exists and its cost.

	A	B	C	D
A	0	2	1	-1
B	2	0	3	5
C	1	3	0	4
D	-1	5	4	0



Expanded list
Fringe list

Sessional syllabus.

BFS.	space complexity $O(b^{d+2})$ time complexity $O(b^{d+2})$.		complete ✓ optimal ×
Uniform cost search	space time	$O(b^{\lceil C^*/\epsilon \rceil})$ —11—	complete if $\epsilon > 0$. optimal ✓
DFS.	time space	$O(b^m)$. $O(bm)$	complete ✓ m finite optimal ×
IDS	time space.	$O(b^{d+1})$ $O(bd)$.	complete ✓ d finite optimal ×
Bidirectional search	time space	$O(b^{d/2+1})$ $O(b^{d/2+1})$	complete ✓ optimal ×
Best First search (h values)	time space		complete ✓ optimal ×
(h values + g values)	A*	time space	complete ✓ optimal ✓
limit	IDA*.	time space $O(bd)$.	complete ✓ optimal ✓
limit (second best)	RBFS.	time space $O(bd)$	complete ✓ optimal ✓
like A*	MBA*	time space M	complete ✓ optimal ✓

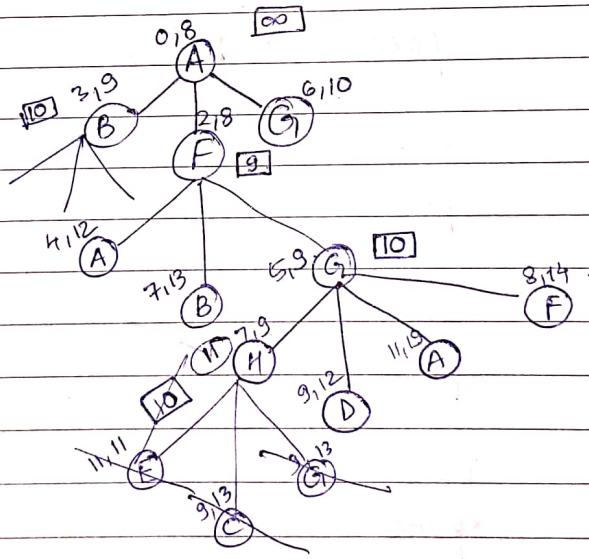
Page No.:
Date:

07/09/19

Local Search Techniques

1) Hill climbing / Gradient descent
2) Random restart hill climbing
3) Local Beam Search
4) Annealing
5) Genetic

RBFS



07/09/19.

Page No.:
Date:

Game playing agents

Environment is entirely observable.

Can we treat games as searching problems ?
(state space search).

2- players zero-sum games
↓
once we reach the end position, we assign some utility to the players.

X O X O X O O O X	X O X O O X X O O	X X O O O X X X O
+1 (-1) A B	-1 (+1) A B	0 (0) A B.

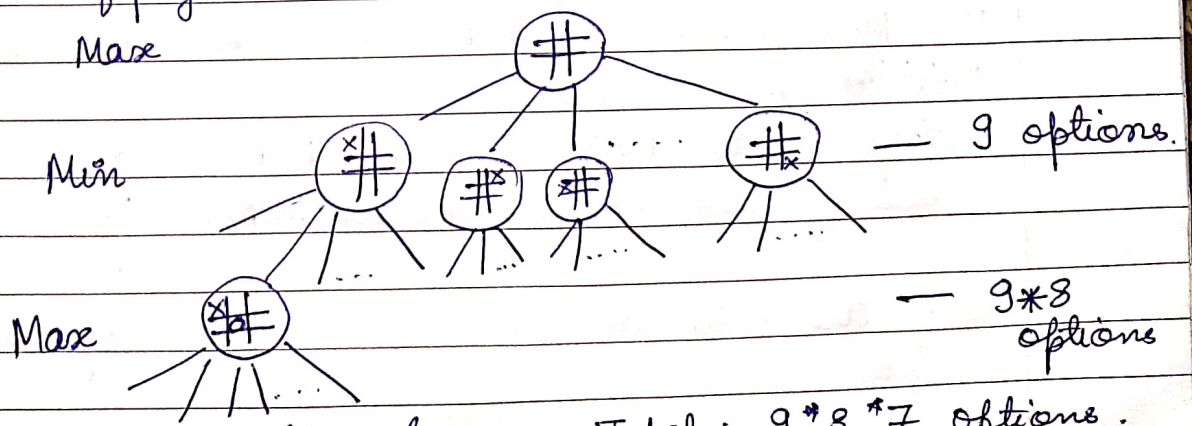
terminal states summation of utility which max player and min player gets is zero.

Move for X → first player.
↓
she will try to achieve goal position where her utility is maximum.
∴ max player.

Move for O → second player
↓
she will try to go for min utility of max player. ∴ min player.

Turn of player next

Tree.



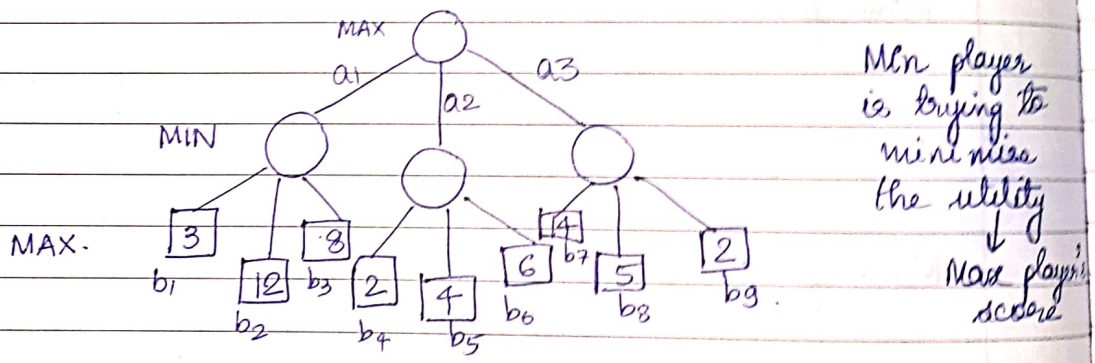
7 options here ∴ Total : 9*8*7 options.

~~At lowest leaf terminal states level~~

At lowest level where all states are terminal states, we will have 9! options.
Some states have +1/-1 and some have zero.

This way, tic tac toe will have a state space.
Memory needed – factorial
BFS – remember every node in list
Breadth first search – lot of memory needed.

DFS used to conserve memory.
But here, the states taken by players depends on what the other player has just played or what she is going to play.



What should MAX play? a1/a2/a3

if MAX takes a1, MIN will take min^m of available utilities i.e 3 (b1).

Similarly a2 (MAX) ⟶ b4 (MIN) gives utility 2.
a3 (MAX) ⟶ b9 (MIN) gives utility 2.

∴ a1 will be the action taken by MAX player.

MAX player at the initial state is doing a state space exploration to find which action to take — a1 or a2 or a3.
Has to take into account that MIN player is also intelligent and her actions should also be explored to decide the MAX player's move.

In TIC TAC TOE however, we do not get utility after single move.
When we reach terminal state, based on player whose ~~s~~ chance is to play just before reaching there → min^m / max^m utility is decided.

★ मतलब ? Then we back off those values till intermediate state and determine its utility

non deterministic other player.
Modelled our program taking other players as an optimal player. → ~~slightly~~ remove the non determinism

It is not possible to model an unoptimum player.
Wayward / non-optimal / drunkard player — X.

This algorithm is called Minimax Search
DFS is used.
Modified DFS where based on player who is playing we take max/min values.

⇒ unroll the recursion
→ we need to explore entire state space to decide the first move by MAX player (xx)

terminal
have zero
space.

depends on
on what

Min player
is trying to
minimise
the utility
↓
Max player's
score

available

utility 2.
utility 2.

AX player.

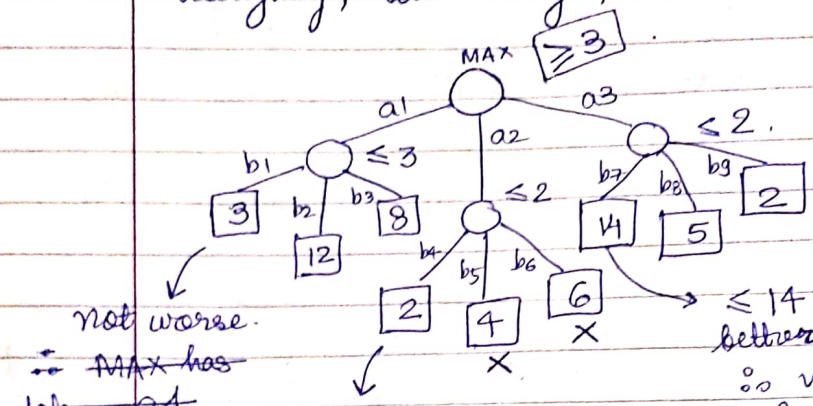
Memory required for TIC TAC TOE

$O(b^m d^m)$. $9^8 . 8^8$.

Page No.: Date: YOUVA

branching factor is 9.

In chess, initially we have 20 options.
Roughly, all things/moves considered – 35 options.



not worse.
∴ ~~MAX has taken a1~~.

here we are already getting a worse value for MAX player. than 3.
∴ we can ignore other two branches and not search that branch.

≤14
better than ≥3 for MAX player
∴ we need to search for other branches – cannot ignore those b8, b9.

This work is done by MAX here – or MIN?

3 is α cutoff (MAX player)
β cutoffs (MIN player)
keep updating

α-β pruning

α-β search

We artificially cutoff the search at certain points

Where do we put the cutoff – at which depth?
– Depends on amount of time we have.

Ludo – non deterministic

Page No.:
Date:

Assignment - II (26th september)

TSP using genetic algorithms.

GA - local search technique
- can be suboptimal as well.

10/09/19

Page No.:
Date:

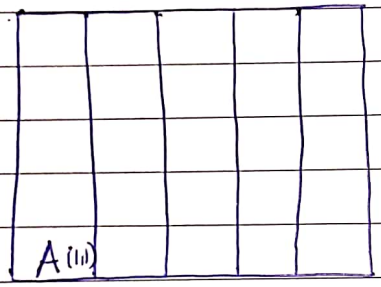
Knowledge Based Agents

$\neg W_{11}$ → Wumpus
$\neg P_{11}$ → Pit
] if both are true then agent is alive in (1,1)

not operator

$\neg B_{11}$ no breeze
$\neg S_{11}$ no stench.

Propositions - true or false. $\neg W_{11}$, $\neg P_{11}$, $\neg B_{11}$, $\neg S_{11}$.
Along with these propositions → need to encode some background knowledge.



These propositions inferred using percepts / sensors.

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

P_{11} ↓ cause to
$B_{12} \wedge B_{21}$ ↓ effect.

	X	Y	X → Y	X ↔ Y
	T	T	T	T
	T	F	F	F
☆	F	T	T	F
	F	F	T	T.

☆ cannot put double implication because of this in $P_{11} \rightarrow B_{12} \wedge B_{21}$.

$B_{11} \leftrightarrow P_{12} \vee P_{21}$ (diagnostic rule)

1) $\neg W_{11}$
2) $\neg P_{11}$
3) $\neg S_{11}$
4) $\neg B_{11}$
5) $B_{11} \leftrightarrow P_{12} \vee P_{21}$

} Knowledge Base KB.

Agents asks questions to itself — α.

$\alpha_1 \xrightarrow{?} \neg P_{12}$

$\alpha_2 \xrightarrow{?} \neg W_{12}$

Entailment (what logically follows).

Iff $KB \models \alpha$, α logically follows, given in KB. (what is)

If knowledge base entails some question or formula. then Models (KB) $\subseteq$ Models (α)

Models of $X \rightarrow Y$ are the rows where $X \rightarrow Y$ becomes true.

Make the truth table for rules till now.
$B_{11}, P_{11}, W_{11}, S_{11}, P_{12}, P_{21}$.

Let's look at scenario of pits and ignore wumpus i.e W_{11}, S_{11} - ignored.

models of KB where KB is true

B_{11}	P_{11}	P_{12}	P_{21}	KB	$\alpha_1 \equiv \neg P_{12}$	$\alpha_2 \equiv \neg P_{21}$
T	T	T	T	F	F	F
T	T	T	F	F	F	T
T	T	F	T	F	T	F
T	T	F	F	F	T	T
T	F	T	T	F	F	F
T	F	T	F	F	F	T
T	F	F	T	F	T	F
T	F	F	F	F	T	T
F	T	T	T	F	F	F
F	T	T	F	F	F	T
F	T	F	T	F	T	F
F	T	F	F	F	T	T
F	F	T	T	F	F	F
F	F	T	F	F	F	T
F	F	F	T	F	T	F
F	F	F	F	T	T	T

For knowledge base to be true, all statements ① – ⑤ in KB must be true.

∴ Models (KB) ⊆ Models (α_1)

⇒ We can conclude that this knowledge base entails α_1.

⇒ There is no pit in cell (1,2).

Models (KB) ⊆ Models (α_2).

∴ KB entails α_2

⇒ There is no pit in cell (2,1)

∴ Agent can go to any cell (1,2) or (2,1).

Suppose it goes to (1,2) and encounters a breeze.
Another diagonastic rule.

⑥ $B_{11} \longleftrightarrow P_{11} \vee P_{13} \vee P_{22}$

P_{11}	B_{12}	P_{13}	P_{22}	KB.	$\alpha_4 \equiv \neg P_{13}$.	$\alpha_5 \equiv \neg P_{22}$.
F	T	T	T	T	F	F
F	T	T	F	T	F	T
F	T	F	T	T	T	F
F	T	F	F	F	T	T
F	F	T	T	F	F	F
F	F	T	F	F	F	T
F	F	F	T	F	T	F
F	F	F	F	T	T	T

⇒ There is a contradiction in first 2 rows for Question $\alpha_4 \xrightarrow{?} \neg P_{13}$.
⇒ For question $\alpha_5 \xrightarrow{?} \neg P_{22}$, contradiction in first and third row.

∴ KB does not entail α_4 and α_5.
→ There can be a pit in both cells P_{13} and P_{22}.
∴ Agent cannot go to (2,2) or (1,3).
⇒ Only option is (1,1).

** This is an Inferencing technique called inference by model enumeration.

Now agent goes back to (1,1)
Moves up to (2,1).
Encounters stench.

Page No.:
Date:

Questions to ask
$\alpha_1 \xrightarrow{?} P_{13}$
$\alpha_2 \xrightarrow{?} \neg W_{31}$

Prove these - as we have with logical reasoning.

Program agent with background knowledge.
Execution of rules done through inference by model enumeration.

Write the rules and ask questions - as a programmer.
Declarative style
Interpreter → carries out the process of inference based on rules we give
It is a reasoner or a theorem prover.

This is exponential.

2^n : where 'n' is no. of variables
↓
size of truth table.

What if we are only given the rules → prove $\neg P_{12}$ without using truth table.

All these rules need to be written for all cells.
Inference - inbuilt
Knowledge - rules.

Axiom formation

Axioms of propositional logic

These are tautologies

1) $\alpha \wedge \beta \equiv \beta \wedge \alpha$
2) $\alpha \leftrightarrow \beta \equiv (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
3) $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$
4) $\alpha \wedge \beta \equiv \alpha$
5) $\alpha \wedge \beta \equiv \beta$
6) $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$
7) $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
8) $\neg(\neg\alpha) \equiv \alpha$
9) $((\alpha \rightarrow \beta) \wedge \alpha) \rightarrow \beta$ Modus Ponens.
10) $\alpha \rightarrow \beta \equiv \neg\beta \rightarrow \neg\alpha$ Contraposition.

★ To prove : $\neg P_{12}$.

Use ② on ⑤

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

Use ⑤

$$P_{12} \vee P_{21} \rightarrow B_{11}.$$

Use ⑩

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

Use ③

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

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

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

$B_{11} \vee \neg P_{12}$.

$T \vee \neg P_{12}$

$\neg P_{12}$

Or.

Use 9. $\neg(P_{12} \vee P_{21})$. (Modus Ponens)

Use 6. $\neg P_{12} \wedge \neg P_{21}$.

Use 1 $\neg P_{12}$.

---

11) $F \vee \alpha \equiv \alpha$
12) $T \wedge \alpha \equiv \alpha$
13) $F \wedge \alpha \equiv F$
14) $T \vee \alpha \equiv T$
15) $\alpha \vee \beta \equiv \beta \vee \alpha$
16) $(\alpha \wedge (\beta \wedge \gamma)) \equiv ((\alpha \wedge \beta) \wedge \gamma)$
17) $(\alpha \vee (\beta \vee \gamma)) \equiv ((\alpha \vee \beta) \vee \gamma)$.
18) $(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$
19) $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$.

Theorem proving

axiom – syntactic translation of a statement from one form to another.

$KB \models \alpha$. (KB follows question or not?)

Find out which statement matches left hand form of an axiom → transform to

right hand form. — and so on. till we get α.
This is a form of a search.
We search for statement and apply axiom

Deduction theorem

Iff $\alpha \models \beta$ then $\alpha \rightarrow \beta$ is a tautology.

Draw a truth table for α, β and $\alpha \rightarrow \beta$ to verify.

Iff $\alpha \not\models \beta$ then $\alpha \wedge \neg\beta$ is unsatisfiable.
↓
no models for this

Transform KB into a form such that resolution can be applied to it. ?.

$KB \wedge \neg\alpha$.

Convert this into conjunctive normal form (CNF).

$C_1 \wedge C_2 \wedge C_3 \wedge \ldots\ldots C_k$. Clauses connected using conjunction.

$C_i \equiv (\neg A \vee B \vee \neg D)$. where each clause is a disjunction of propositions.

KB.

CNF $\neg B_{11}$ | $\alpha \equiv \neg P_{12}$.

X CNF $B_{11} \leftrightarrow P_{12} \vee P_{21}$
↳ convert to CNF.

1) Remove $\leftrightarrow$, if present.

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

2) Remove $\rightarrow$

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

3). Use D' Morgans Law.

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

4) Distribute $\vee$ over $\wedge$.

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

C_1 : $\neg B_{11} \vee P_{12} \vee P_{21}$
C_2 : $\neg P_{12} \vee B_{11}$
C_3 : $\neg P_{21} \vee B_{11}$
C_4 : $\neg B_{11}$
C_5 : $\neg P_{12}$.

KB converted to CNF using these rules.

Invalid

Resolution rule

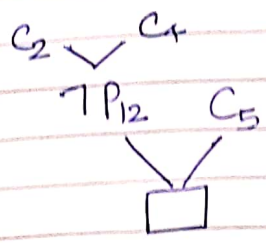
We look for two clauses < one form / -ve form.

$\neg a \vee (\neg b) \vee c$, $(b) \vee \neg e$

$\therefore \neg a \vee c \vee \neg e$.

Inferencing mechanism of propositional logic

One of the clauses comes out to be null clause or false when we keep on resolving



once sents converted to set of clauses - check for resolution

Keep on resolving until we reach null clause.

↓

This tells that KB entails α.

If we do not get a null clause

→ KB does not entail α.

Propositional Inference — NP complete.

We tell the agent what we know.
Based on inference technique, actions are decided.

Page No.:
Date:

12/09

12/09/19

$KB \models \alpha$

(1) Inference by enumeration

Models (KB) $\subseteq$ Models (α).

(2) Theorem proving by resolution

$KB \wedge \neg\alpha$.

$x_i = \neg y_j$

(C_1) $x_1 \vee \dots \vee x_{i-1} \vee x_i \vee x_{i+1} \dots \vee x_n$, $y_1 \vee \dots y_j \vee \dots \vee y_m$ (C_2)

$x_1 \vee x_2 \dots \vee x_n \vee y_1 \vee y_2 \dots \vee y_m$

Soundness

If x_i is false, C_1 is known to be true

$\therefore$ $x_1 \vee \dots x_{i-1} \vee x_{i+1} \dots x_n$ is true.

$\neg y_j$ = false $\therefore$ y_j is true $\Rightarrow$ C_2 is true.

$\Rightarrow$ $C_1 \vee C_2$ is true.

Complete, Refutation completeness.

if α entailed by KB : null clause, algo stops.
if α not entailed by KB : fixed point reached
- no further resolution

What all can be entailed by KB ?

→ cannot be answered by resolution

$KB \wedge \neg\alpha \equiv$ CNF.

Deduction theorem : $KB \wedge \neg\alpha$ is unsatisfiable if KB entails α.

If α is entailed by KB,

$C_1 \wedge C_2 \ldots \wedge C_k$ is false.

∴ Atleast one of $C_1, C_2 \ldots C_k$ is false.

Can we determine falsehood of some proposition beforehand instead of going for brute force in enumeration

DPLL algorithm (1960's). Inference by Elimination

Heuristics proposed by DPLL.

1) Unit clause - only one literal / proposition

KB - true, unit clause literals assigned true. or give them a value such that there is a possibility of existence of models for that KB.

$x_i \rightarrow$ T to x_i

$\neg x_i \rightarrow$ F to x_i

2) Pure symbol / pure literal

3)

pure : all clauses wherever it occurs → it is always in either +ve sense or all in -ve sense.

If a pure literal exists give it one value and check for models. Then give the other value and check for models.

3) Most frequent literal : symbol / literal occurring most frequently (polynomial time wrt. no. of clauses).
↓
+ve / -ve.

Say x_i : most frequent literal.
+ve sense : 7 times
-ve sense : 3 times

When some clause becomes true, we can omit it from our consideration of
$C_1 \wedge C_2 \wedge \dots \wedge C_k$.

C_i : $x \vee \neg y \vee z \vee \neg k \vee p$.
C_i can be omitted from consideration ← if any one becomes true → whole clause becomes true.

∴ give x_i value true because more clauses may become true because of that and reduce the number of clauses we need to check.

4) Unit propogation : trying to decide value of x.
say x : True.
C_i becomes a unit clause.
↓
apply first heuristic here.
y = true.

C_i : $\neg x \vee y$
C_j : $\neg z \vee y$

assign T to y ← both clauses are removed from consideration
C_j becomes T.

If $C_k \equiv \neg y \vee w$.

y becomes true ∴ $\neg y \equiv$ false

∴ $C_k \equiv w$ (unit clause)

$(w = T)$.

∴ In this way, unit clause is propogated.

Example :

C_1 : $\neg a \vee \neg b \vee c$

C_2 : $\neg a \vee \neg b \vee \neg c$

C_3 : $a \vee \neg b \vee \neg f \vee \neg g$

C_4 : $\neg a \vee \neg d \vee b$

C_5 : $\neg a \vee e$

C_6 : $e \vee f \vee g$

C_7 : $b \vee c \vee g$

At most 1 +ve literal present in a clause

Cannot use forward chaining or backward chaining — C_6, C_7.

1) no unit clause

2) pure literal : e.

$e = T$ remove C_6 and C_5 from consideration

pure literal : d.

$d = F$ remove C_4.

pure literal : f

$f = F$ remove C_3.

pure literal : g

$g = T$ remove C_7.

left with C_1 and C_2, a and b are now pure.

[∴ $\neg KB \wedge \neg\alpha$ is not false
α is not entailed by KB]

pure literal : a
$a = F$ remove C_1, C_2.

We are left with null ~~now~~
∴ α is entailed by KB.

But here we found a model.
↓
∴ KB does not entail α.

Model : A: D: G:
B: E:
C: F:

$KB \wedge \neg\alpha \longrightarrow$ we constructed CNF.
We found a model for CNF
∴ KB does not entail α.
because $KB \wedge \neg\alpha$ is satisfiable.

Resolution : null clause ⇒ unsatisfiable
⇒ KB entails α.

Q.

C_1 : $\neg x_1 \vee x_2$
C_2 : $\neg x_1 \vee x_3 \vee x_9$
C_3 : $\neg x_2 \vee \neg x_3 \vee x_4$.
C_4 : $\neg x_4 \vee x_5 \vee x_{10}$
C_5 : $\neg x_4 \vee x_6 \vee x_{11}$
C_6 : $\neg x_5 \vee \neg x_6$.
C_7 : $x_1 \vee x_7 \vee \neg x_{12}$
C_8 : $x_1 \vee x_8$
C_9 : $\neg x_7 \vee \neg x_8 \vee \neg x_{13}$
C_{10} : $x_9 \vee x_{10} \vee x_{11} \vee \neg x_1$.

x_4 : 5 times most frequent.
$\neg x_4$: 3
x_4 : 2
∴ $x_4 = F$.

$x_4 = F$ ⇒ C_1, C_2, C_{10} are removed.
C_7, C_8 ∄ x_1 removed.
C_8 becomes unit clause
⇒ $x_8 = T$.
C_9 ∄ remove $\neg x_8$.

Page No.:
Date:

Now x_4 occurs most frequently
$\neg x_4$: 2 times
x_4 : 1 time
$x_4 = F \Rightarrow C_4, C_5$ are removed
$\Rightarrow C_3$ ~~if~~ remove x_4.

Now, x_7 occurs 2 times
$x_7 = T \Rightarrow C_7$ is removed
$\Rightarrow C_9$ ~~if~~ x_7 removed.
C_7 becomes unit clause
$\therefore x_{13} = F \Rightarrow C_9$ is removed.

Now, $x_2 = F \Rightarrow C_3$ is removed.
$x_5 = F \Rightarrow C_6$ is removed.

Model :

(3)

A model exists $\therefore \alpha$ is not entailed by KB.
↓
7 assignments

Ex)

C_1	$x \vee y \vee z$	
C_2	$x \vee y \vee \neg z$	y : 6 times
C_3	$x \vee t \vee \neg y$	$\neg y$: 3 times
C_4	$\neg t \vee x$	y : 3 times
C_5	$\neg x \vee y$	
C_6	$\neg y \vee z$	$y = T$. (say).
C_7	$\neg y \vee \neg z$	remove C_1, C_2, C_5.

C_7 becomes unit clause, $z = F$
but C_6 becomes false.

This contradiction tells that no model exists.
Now check for $y = F$.
remove C_3, C_6, C_7.
C_5 becomes unit clause $\therefore x = F$.
remove C_5.
C_1, C_2 are unit clauses with z and $\neg z$.
$\therefore$ contradiction.

** enumeration carried out recursively here.

⇒ No model exists for all possibilities.
⇒ α is not entailed by KB.

** check for all possibilities of all literals.

Assignment 3 : Implementation of DPLL.

③ Local Search Technique. to find out satisfiability
↓
Incomplete – model may exist, but even after max^m no. of iterations – we don't get a model.
– Randomness helps.

Walk SAT . NP – verified in polynomial time

$C_1 \wedge C_2 \ldots \wedge C_k$ trying to find a model for this (k clauses).
n : no. of literals
each C_i can be of any length.

1) Randomly assign T/F values to the literals.
2) check if this is a solution
(every clause should be true) – polynomial time

3) If yes, stop. α is not entailed by KB.

4) If not, pick a random unsatisfied clause say C_i : $\alpha \vee \neg y \vee z \vee \neg p \vee \neg w$.

C_i : false. Check out why it is false.
→ all the literals are false (OR).

Flip any one literal — which one ?
↓ make one proposition ~~true~~ — clause ~~becomes true~~.
↓ The literal which occurs max^m no. of times in unsatisfied clauses in the same sense.

5) Flip the value of that ~~clause~~ literal such that it results in maximum number of ~~satisfied~~ clauses. ~~with~~.
↓
flip this value with a probability 'p'.
flip any value with probability '1-p'.

6) With probability (1-p), flip the value of a randomly chosen literal.

7) reiterate from step (2).

Run for a number of iterations — if we find a solution - okay.
- if we don't find a solution, it has failed.
↓ Walk SAT.

If walkSAT succeeds - KB does not entail α.

1)
2)
3)
4)
16)

one in

Horn type - restricted form.
- P type.

Inference techniques for propositional logic.
- DPLL
- theorem proving using resolution
- Walk SAT.

Propositional logic (Wumpus World).

- does not have quantifiers $\forall$, $\in$
- write rules for all cells.

			(4,4)
(3,1)	(3,2)		(3,4)
(2,1)	(2,2)	(2,3)	(2,4)
(1,1)	(1,2)	(1,3)	(1,4)

1) $B_{11} \leftrightarrow P_{12} \vee P_{21}$ diagonistic rule.
2) $B_{12} \leftrightarrow P_{13} \vee P_{22} \vee P_{11}$
3) $B_{13} \leftrightarrow P_{12} \vee P_{23} \vee P_{14}$
4) $B_{22} \leftrightarrow P_{21} \vee P_{32} \vee P_{23} \vee P_{12}$.
16) B_{44}

To state that only one wumpus is present.

$W_{11} \vee W_{12} \vee W_{13} \ldots\ldots W_{44}$ → atleast one wumpus exists

KB ∴ this rule ⇒ it is true
⇒ atleast one of W_{ij} is true.

$$\{(W_{11} \wedge \neg W_{12} \wedge \neg W_{13} \ldots \neg W_{44}) \vee (\neg W_{11} \wedge W_{12} \wedge \ldots \neg W_{44}) \vee$$
$$(\neg W_{11} \wedge \neg W_{12} \wedge W_{13} \wedge \ldots \neg W_{44}) \vee \ldots\ldots (\neg W_{11} \wedge \neg W_{12} \ldots \wedge W_{44})\}$$

one rule. to prove only one wumpus is present in KB.

How to keep track of agent?

$L_{11} \wedge Right \rightarrow L_{12}$.

this does not say that agent is not present in (1,1) any more.

Omit some rules for boundary cases.

Page No.:
Date:

∴ Include a factor of time with all rules.

$L_{11}^{0} \wedge right^{0} \rightarrow L_{12}^{1} \wedge \neg L_{11}^{1}$

O_{t1} : time instant

17/09

Write for all cells — right move, up move, left move, down move.
(omit Boundary cases).

How many such rules for various time steps?
Add rules only for $t = 0$.
If at $t=1$, we don't reach goal, write rules for $t = 1$.

↓

$t = 1$: right / left / up / down rules for all cells.

∴ KB keeps increasing with time.

Eg:

If we had quantifier : $\forall_{x,y,t} \; L_{x,y}^{t} \wedge right^{t} \rightarrow L_{x,y+1}^{t+1} \wedge \neg L_{x,y}^{t+1}$

↓ for all time steps

— 1 rule for one cell's right move.

↓

first order logic → more powerful than propositional logic.

Inferencing in first order logic is semi decidable / undecidable.

17/09/19

First Order Logic

1) constants
2) propositions
3) connectives
4) objects (of the real world – state something about them)
5) relations / predicates (b/w the objects)
6) functions
7) quantifiers ($\forall, \exists$).

relations – unary / binary
Eg: student (x) : x is a student
student of (x, y) : x is a student of y.
(x ↓ person, y ↓ organisation)

functions – father (x)
function that maps to an entity in the domain that we have.

$\forall x, y, t \quad At(agent, [x, y])^t \wedge right^t \longrightarrow At(agent, [x+1, y])^{t+1}$.

agent – constant
$[x, y]$ – syntactic sugar
$\forall$ universal quantifier

$\exists$ – existential quantifier

$\exists x \; Person(x)$.

VNIT(x).

Page No.:
Date:

1. All students of VNIT are smart.

Predicates : Student (x)
College (y).
Studies_in (x, y)
Smart (x)

x, y, ... small case — variables.
something that starts with capital — function / predicate / constant.

subject — ?.

should be true for all x in domain ←

$\forall x$ Student(x) $\wedge$ College(VNIT) $\wedge$ Studies_in (x, VNIT) $\rightarrow$ Smart(x)

If we change $\rightarrow$ as $\wedge$:

$\forall x$ Student (x) $\wedge$ College (VNIT) $\wedge$ Studies_in (x, y) $\wedge$ Smart (x) $\rightarrow$ in KB, every term has to be true.

↓

means everybody is a student, everybody studies in VNIT, everybody is smart

Truth table

2) Some student at VNIT is lazy.

$\exists x \; Student(x) \wedge Studies\ in(x, y) \wedge Lazy(x)$

If we replace $\wedge$ with $\rightarrow$, draw truth table.

Page No.:
Date:
YOUVA

Brother (x,y)
Sister (x,y)
Male (x)
Female (x)
Parent (x,y)

1) A sibling is either a brother or a sister

$\forall x,y$ Brother $(x,y) \vee$ Sister $(x,y) \leftrightarrow$ Sibling (x,y).

2) Mother is one's female parent

$\forall x,y$ Mother $(x,y) \leftrightarrow$ Parent $(x,y) \wedge$ Female (x)

3) Cousin is a child of parent's sibling

$\forall x,y,z,w$ Parent $(x,y) \wedge$ Sibling $(z,x) \wedge$ Parent $(z,w) \leftrightarrow$ Cousin (w,x).

4) Ram has atleast two brothers.

$\exists x,y$ Brother $(x, Ram) \wedge$ Brother (y, Ram).

↓

x is brother of Ram.

Equality in first order logic checks whether object LHS refers to RHS object or not

$\therefore \exists x,y$ Brother $(x, Ram) \wedge$ Brother $(y, Ram) \wedge \neg(x=y)$.

5) Ram has exactly two brothers.

¬(there exists a third brother) to AND have with above stmt.

$(\exists x,y \text{ Brother}(x, \text{Ram}) \wedge \text{Brother}(y, \text{Ram}) \wedge \neg(x=y)) \wedge$
$(\neg(\exists z \text{ Brother}(z, \text{Ram}) \wedge \neg(x=z) \wedge \neg(y=z)))$

Q. Function — $\text{Succ}(x) \longrightarrow x+1$.

KB. { 1) WholeNum (0).
2) $\forall n$ WholeNum $(n) \longrightarrow$ WholeNum (Succ(n)).

$\alpha \longrightarrow$ WholeNum (99) $\longrightarrow$ T/F.

Does KB entail α ?

Q. Person (x), Brave (Ram)
Person (Ram), Person (Laxman), ~~King(x)~~
$\forall x$ Person $(x) \wedge$ Brave $(x) \longrightarrow$ King (x).

$\exists p$ King (p). ?

If we return $\{p/\text{Ram}\}$. Substitution list

☆☆ Our inference can return a substitution list.

$\exists_p$ person (p) ?

$\{p/\text{Ram}, p/\text{Lakshman}\}$ — substitution list

Prologue — facts and rules.
↓
in reverse order

Quantifier $\forall$ assumed to be Implicit.

King $(x) \rightarrow$ Person (x), Brave (x)

When question is asked — $\exists$ is implicit.

18/09/19.

Page No.:
Date:

1) $\forall x$ King $(x) \wedge$ Greedy $(x) \rightarrow$ Evil (x)
2) King (John) (Fact)
3) $\forall x$ Greedy (x).
4) Brother (John, Richard) [Grounded terms].

Entailment

Model (Domain)
Interpretation (which object are we referring to)

Q. Evil (John). Does KB entail α ?

Propositionalization → convert first order rules to propositions.

1) Universal instantiation

Whenever we have a universal quantifier with a variable, we'll replace the variable with all constants/entities of the domain.

King (John) $\wedge$ Greedy (John) → Evil (John)
King (Richard)
King (Jack)

↓

These are now propositions and we can find entailment with methods as discussed.

Greedy (John)
Greedy (Richard)
Greedy (Jack).

(1) and (3) can now be removed from KB.

2) Existential Quantifier removal Skolemisation

When a rule has existential quantifier, we replace the variable with some constant 'K'.

$\exists y$ Greedy (y) $\Rightarrow$ Greedy (K)

↓ some K that exists in domain.

First order KB is converted to propositional KB

Problem — When we have function symbols are present in KB

↓ maps onto a unique object in domain by taking a parameter.

5) Father (John)

1) Universal Instantiation

Replace variable with function symbols also.

King (Father (John)) $\wedge$ Greedy (Father (John)) $\rightarrow$ Evil (Father (John))

Same for Father (Richard)
Father (Jack)
Father (Father (Richard))
⋮

Father (John) → maps to some objects/constants in domain.

KB now becomes Infinite

Agar Father (x) hota with some quantifier — first that will be propositionalised and then used here.

WholeNum (0)
$\forall n$ WholeNum (n) → WholeNum (Succ(n))

First we will replace 'n' with constant in KB.
i.e 0.
Apply successor function again

If we do not have function symbols, first order logic becomes decidable.
~~Nat~~ WholeNum (3.14) ↙
else it becomes infinitely executable.

Also, to reduce KB, we can substitute only those constants present in question
i.e Evil (John) ?
↓
~~substitute only John in (1) and (3).~~

(3) $\exists x$ Evil(x) $\{x/John\}$.

Generalised Modus Ponens

King(x) ∧ Greedy(x) → Evil(x)
King (John)
Greedy (y).

King ( ) → We have King (John). ∴ x/John.
↓
then King(x) will become true.

x/y → Greedy (y) becomes true.
↓
Greedy (John) is true.

∴ King (John) ∧ Greedy (John).

∴ LHS becomes true ⇒ RHS should be true.
Evil(John) → true.

Evil(x) x/John

$$\frac{p_1 \wedge p_2 \wedge \cdots p_n \rightarrow q, \quad p_1', p_2', \cdots p_n'}{q/\theta}$$

Whether q is entailed and by what substitution

if $p_1/\theta = p_1'/\theta$
$p_2/\theta = p_2'/\theta$
$\vdots$
$p_n/\theta = p_n'/\theta$

24/9/19

If p1 matches with p_1' under substitution θ and p2
⋮
then q under substitution θ is true.

Eg: $\forall x$ King(x) ∧ Greedy(x) → Evil(x)
$\forall y$ Greedy(y)
King(John)

p_1: King(x) p_1' King(John)
p_2: Greedy(x) p_2' Greedy(y)
q: Evil(x) x=y= John.

p_1/John = p_1'/John
p_2/John = p_2'/John

∴ q/John is true ⇒ Evil(John).

Page No.:
Date:
YOUVA

** Horne clauses only.

Find out substitutions under which (p_1, p_1'), (p_2, p_2') .... will match with each other.

Unification : String matching $O(n^2)$
(Finding substitutions)

$\forall x$ Knows (John, x) , Knows (John, Jack).

x = Jack These two terms will unify with each other under substitution x = Jack.

$\forall x$ Knows (John, x) , $\forall y$ Knows (y, Richard)

x = Richard.
y = John

Unification algorithm returns a substitution list for which these terms will match.

$\forall x$ Knows (John, x) , $\forall y$ Knows (y, Father (y))

x = Father (John)
y = John.

$\forall x$ Knows (John, x) , $\forall y \forall z$ Knows (y, z).

* y = John
* x = z. (Not saying what value x and z should have).

general unifier.

$\forall x$ Knows (John, x) , $\forall x$ Knows (x, Richard)

x = Richard x = John.

Not possible for two values of x. are not allowed.

∴ Failure returned by unification algorithm

It is better to use different variable for every rule / formula / sentence.

Paragraph

⇒ It is a crime for an American to sell weapons to hostile nations.

⇒ Country. Nono has some missiles (America is hostile to Nono)

⇒ All the missiles owned by Nono were sold to it by Col. West who is an American.

Q. Is Col. West a criminal ?

Predicates

American - Unary
Weapon - Unary
Hostile - Binary
Sells - Ternary
Criminal - Unary

1) $\forall x, y, z$. American(x) $\wedge$ Weapon(y) $\wedge$ Sells(x, y, z) $\wedge$ Hostile(America, z) $\longrightarrow$ Criminal(x).

Enemy (x, y) - Binary [Hostile भी use कर सकते थे].
Has (x, y) - Binary
Missile (x) - Unary.

2) Enemy (America, NoNo)

3) $\forall x', y'$ Enemy $(x', y') \rightarrow$ Hostile (x', y')

4) $\exists m$ Has (Nono, m) $\rightarrow$ Missile (m).

5) $\forall m'$ Missile $(m') \rightarrow$ Weapon (m').

6) American (Col. West).

7) $\forall m''$ Missile $(m'') \wedge$ Has (Nono, m'') $\rightarrow$
Sells (Col. West , m'', Nono) $\wedge$

(4) – Existential Quantifier needs to be removed.
4a) Has (Nono, M).
4b) Missile (M).

Q. Is Col. West Criminal ? Criminal (Col-West)?

(1). Match for x = Col. West.
But no substitution for y.

(2) Grounded

(3). Use GMP on (3) and (2) with substitution
x' = America We get
y' = Nono (8) Hostile (America, Nono)

(4) Grounded to 4a, 4b.

(5). Use GMP on (5) and 4(b) with substitution
m' = M. We get
(10). Weapon (M).

(6) (7) Use GMP on (7) and (4a) and (4b).
Substitute $m'' = M$.
(9) We get Sells (Col. West, M, NoNo).

(6) Grounded.

One pass done. Now we go to second pass because question not yet answered.

(1) Use GMP on (1) and (6), (9), (10), (8).
x = Col. West
y = M
z = NoNo
We get
(11) Criminal (Col. West)
↓
This was the query.

∴ KB entails α.

repeated application of GMP.

This process is called forward chaining.
Forward chaining can answer general questions also.

Eg: Is there a criminal?
$\exists x''$ Criminal (x'').

Inference engine will return an answer that $x'' = $ Col. West is a substitution with which query is true.
↓
can be a substitution list as well.

Next: Limitations of forward Chaining
Backward chaining – used by prologue.

25/9/19

In the k^{th} iteration we should consider those rules added in $(k-1)^{th}$ iteration (pass)

When we look at grounded rules, we should mark up rules that might be fired in the next pass. Also do this whenever new rule is added to KB.

Consider the conjunction having least no. of options first.

$A(x) \wedge B(x) \rightarrow C(x)$ Consider $A(x)$ first.

$A(x)$: 2 options; $B(x)$: 3 options

Remember partial matches in memory — to avoid unnecessary matchings for already matched conjunctives in the successive passes

Semi decidable - Inference for first order logic
If no proof - might loop infinitely.

How do we do matching?

- One table for each predicate / relation

Eg: Student (Institution, student-name)

Indexing that table on the basis of one of the parameters / both parameters

forward chaining – All possible questions that might be entailed by KB are found out.

Backward Chaining

(Find only whether α is entailed by KB)

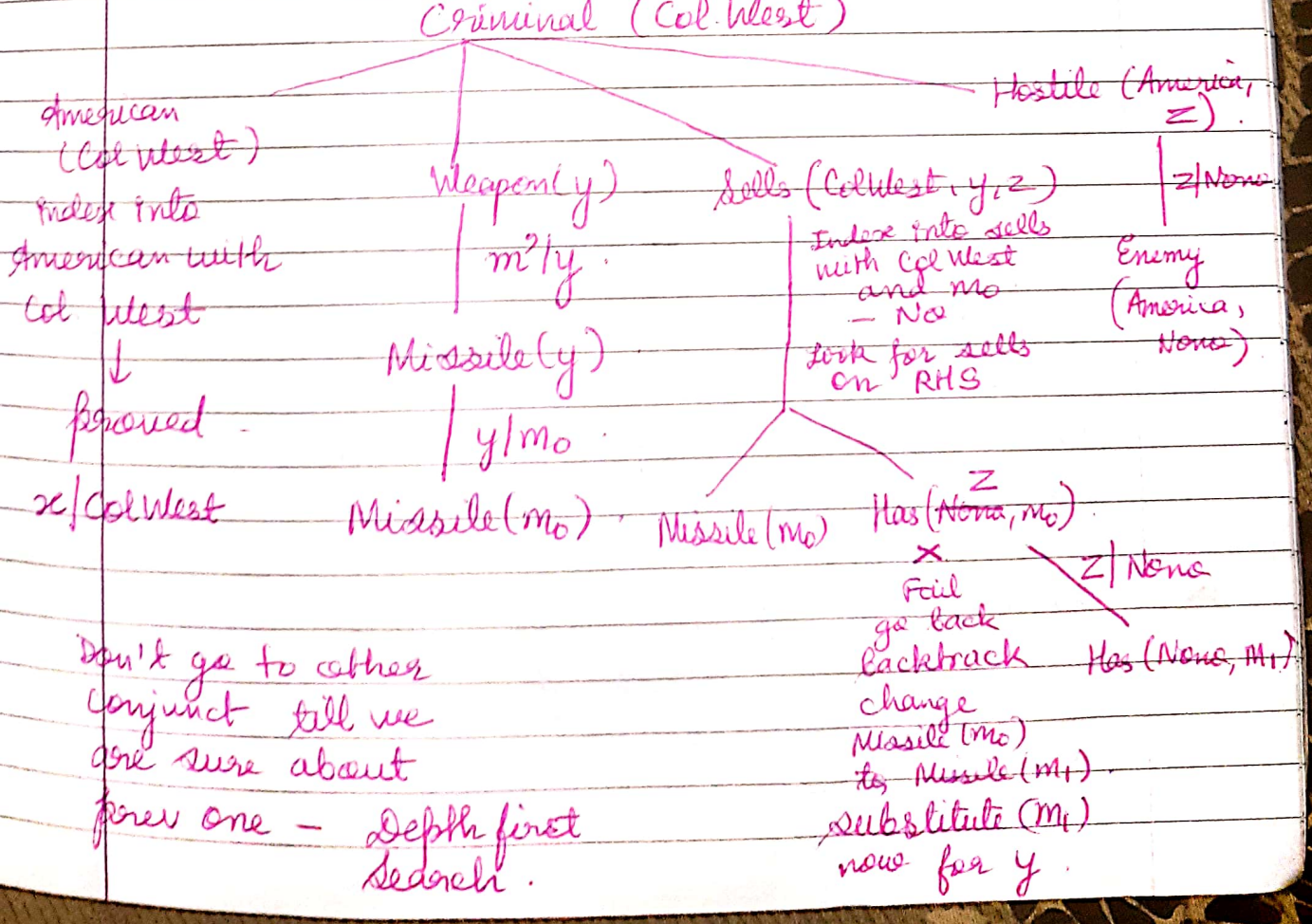
Start from the question
Search in KB – find if already entailed.

Criminal (Col West)

If we don't get true, look for Criminal( ) ← relation / predicate

Hashing key – Col West
Search for predicates with Col West as index.

Look for criminal(x) → write precedents.



Don't go to other conjunct till we are sure about prev one – Depth first search.

Page No.:
Date:

01/10

Till we reach grounded predicate.
If we don't get a substitution, go to prev branch by backtracking and try some other substitution.

Q) $\exists x'$ Criminal (x').

Criminal (x')
x'/Colwest
American (Col. West)

If there are 1000 Americans,

1) Try out all options
2) Conjunct ordering
consider conjuct with least no. of options.

Prologue Writing

Prologue uses B/W chaining.

Q) criminal (x)
$x = \{$ Col. West $\}$.

Both F/W and B/W works for
1) implications and facts
2) all in +ve sense
3) RHS - single
4) LHS - conjunction connected

$\neg$ American $(x) \vee \neg$ Weapon $(x) \vee \neg$ Sells $(x, y, z) \vee$
$\neg$ Hostile (American, z) $\vee$ Criminal (x)

★★ Monotonic - propositional, first order
Non monotonic reasoning

01/10/19.

Resolution in FoL

$$\frac{l_1 \vee l_2 \vee \dots l_n \;,\; k_1 \vee k_2 \vee \dots k_m}{l_1 \vee l_2 \vee l_3 \dots l_n \vee k_1 \vee k_2 \dots \vee k_m \,|\, \theta} \quad \text{remove } l_i \text{ and } k_j$$

$$l_i|_\theta = \neg k_j|_\theta$$

$\forall x$ At (VNIT, x) $\rightarrow$ Smart(x)
At (VNIT, Ram)

$$\frac{\forall x \;\; \neg At(VNIT, x) \vee Smart(x) \;,\; At(VNIT, Ram)}{Smart(Ram)}$$

Substitute x = Ram,

$\neg$At (VNIT, Ram) = $\neg$At (VNIT, Ram)
$\neg$At (VNIT, x)$|_{Ram}$ = $\neg$At (VNIT, Ram)

∴ Smart (Ram) is inferred.

Everyone who loves all animals is loved by someone

$\forall x \{ [\forall y$ Animal(y) $\rightarrow$ Loves (x,y)$] \rightarrow \exists z$ loves (z,x)$\}$

(scope of y: Animal(y) $\rightarrow$ Loves (x,y); scope of z: loves (z,x); scope of x: the whole formula)

Convert to CNF.

1) Eliminate $\rightarrow$, $\leftrightarrow$

$\forall x \big(\neg [\forall y \;\; \neg$Animal($y$) $\vee$ Loves (x,y)$] \vee \exists z$ loves (z,x)$\big)$

Page No.: Date: YOUVA

2. Push ¬ inside.

$\neg \forall x\ p \rightarrow \exists x\ \neg p$

$\neg \exists x\ p \rightarrow \forall x\ \neg p$

$\forall x \left[\exists y\ \neg(\neg Animal(y) \vee loves(x,y)) \vee \exists z\ loves(z,x)\right]$

$\forall x \left\{\left[\exists y\ Animal(y) \wedge \neg loves(x,y)\right] \vee \exists z\ loves(z,x)\right\}$

3. Remove existential quantifier using skolemisation.

$\forall x\ [Animal(k_1) \wedge \neg loves(x, k_1)] \vee loves(k_2, x)$

This is not right.

$y \mid F(x)$, $z \mid G(x)$. $F(x)$, $G(x)$ do not belong to KB.

dependent on x.

it's not a constant person but somebody dependent on x.

replace skolem vars not with constants but with functions.

$F(x)$, $G(x)$ skolem functions instead of skolem constants and arguments of skolem functions will be variables present in that statement.

$\forall x\ [Animal(F(x)) \wedge \neg loves(x, F(x))] \vee loves(G(x), x)$

4) Distribute $\vee$ over $\wedge$.

C_1

$\forall x \; [Animal(F(x)) \vee Loves(G(x), x)] \wedge$
$[\neg Loves(x, F(x)) \vee Loves(G(x), x)]$

C_2.

2) Anyone who kills an animal is loved by no one

$\forall x \; [\exists y \; Animal(y) \wedge Kills(x, y)] \rightarrow \forall z \; \neg Loves(z, x)$

1) ~~$\forall x \; \exists y \; \neg Animal(y)$~~

$\forall x \; \neg [\exists y \; Animal(y) \wedge Kills(x, y)] \vee \forall z \; \neg Loves(z, x)$

2) $\forall x \; \forall y \; [\neg Animal(y) \vee \neg Kills(x, y)] \vee \forall z \; \neg Loves(z, x)$

$\forall x \; \forall y \; \forall z \; \neg Animal(y) \vee \neg Kills(x, y) \vee \neg Loves(z, x)$

C_3 : $\neg Animal(y') \vee \neg Kills(x', y') \vee \neg Loves(z', x')$.

3) Jack loves all animals

C_3 : $\neg Animal(y'') \vee Loves(Jack, y'')$.

4) Either Jack or curiosity killed the cat named Tuna

C_4 : $Cat(Tuna)$.

Page No.:
Date:

C_5 : Kills (Jack, Tuna) V Kills (Curiosity, Tuna)

C_6 : $\neg$Cat(y''') V Animal(y''')

Q. Did curiosity kill Tuna ?

C_7 : α : $\neg$Kills (Curiosity, Tuna)

Prologue only supports Horn clauses.

Assignment III

Input a set of propositional clauses.
Find out if a model exists for the clauses using DPLL Algo.

Eval date : 24/10/19.

03/10

03/10/19

C_1 : Animal ($\exists$ F(x)) $\vee$ Loves ($G_1(x)$, x)

C_2 : $\neg$Loves (x, F(x)) $\vee$ Loves ($G_1(x)$, x)

C_3 : $\neg$Animal (y) $\vee$ $\neg$Kills (x_1, y) $\vee$ $\neg$Loves (z, x_1)

C_4 : $\neg$Animal (x_2) $\vee$ Loves (Jack, x_2)

C_5 : Kills (Jack, Tuna) $\vee$ Kills (curiosity, Tuna)

C_6 : $\neg$Cat (x_3) $\vee$ Animal (x_3)

C_7 : Cat (Tuna)

C_8 : $\neg$ Kills (Curiosity, Tuna)

C_6 , C_7 ; x_3 | Tuna

C_9 : Animal (Tuna)

C_5 , C_8

C_{10} : Kills (Jack, Tuna)

C_{10} , C_3 ; x_1 | Jack, y | Tuna

C_{11} : $\neg$Animal (Tuna) $\vee$ $\neg$Loves (z, Jack)

C_{11} , C_9

C_{12} : $\neg$Loves (z, Jack)

C_{12} , C_4 ; x | Jack, x_2 | F(Jack)

C_{13} : $\neg$Animal (F(Jack)) $\vee$ Loves (G_1(Jack), Jack).

C_1 , C_{13} ; x | Jack

C_{14} : Loves (G_1(Jack), Jack).

C_{14} , C_{12} ; z | G_1(Jack).

Null

$\therefore$ α is entailed by KB.

$\Rightarrow$ Curiosity killed Tuna.

If question is : Who killed Tuna ?

$$\neg [\underbrace{\exists x_3 \text{ Kills } (x_3, \text{Tuna})}_{\alpha}]$$

$$= \forall_{x_3} \neg \text{Kills } (x_3, \text{Tuna})$$

$\{x_3 / \text{curiosity}\}$ — then this α is entailed.

C_5 C_3

$x_3/\text{curiosity}$

If we substitute x_3 with Jack,

Some heuristics can be used to search for two clauses to resolve.

Go for a shorter clause over a longer one.

goes into infinite loop when α is not entailed.

Q. $\forall x$ American $(x) \wedge$ Weapon $(y) \wedge$ sells $(x, y, z) \wedge$ Hostile (America, z) $\rightarrow$ Criminal (x)

C_1 : $\neg$American$(x) \vee \neg$Weapon$(y) \vee \neg$sells (x, y, z) $\vee \neg$Hostile (America, z) $\vee$ Criminal (x)

C_2 : Owns (Nono, M) $\quad \exists x_1$ Owns (Nono, x_1) $\wedge$ Missile (x_1).

C_3 : Missile (x_2) M).

$\forall x_2$ Missile $(x_2) \wedge$ Owns (Nono, x_2) $\rightarrow$ Sells (Col West, x_2, Nono)

C_4 : $\neg$Missile $(x_2) \vee \neg$Owns(Nono, x_2) $\vee$ Sells (Col West, x_2, Nono)

$\forall x_3, y_3$ Enemy $(x_3, y_3) \rightarrow$ Hostile (x_3, y_3)

C_5 : $\neg$Enemy $(x_3, y_3) \vee$ Hostile (x_3, y_3).

C_6 : Enemy (America, Nono)

C_7 : American (Col. West)

α : $\exists x_4$ Criminal (x_4)

$\neg\alpha$: C_8 : $\neg$Criminal (x_4).

C_6 , C_5 : x_3 | America, y_3 | Nono.

C_9 : Hostile (America, Nono).

C_2 , C_4 : x_2 | M.

C_{10} : $\neg$Missile (M) $\vee$ Sells (Col. West, M, Nono).

C_{10} , C_3.

C_{11} : Sells (Col. West, M, Nono)

C_1 , C_{11} : x | Col West, y | M, z | Nono.

$\neg$American (x) $\vee$ Weapon (M) $\vee$ $\neg$Hostile (America, Nono) $\vee$ Criminal (x).

C_{13} : $\neg$Missile $(x_5) \vee$ Weapon (x_5).