8/8/19

Constraint Satisfaction Problem (CSP)

Not all AI problems are similar to 8-tile puzzle.

They search for the goal state but, with certain constraints which need to be taken care of.

CSP is a kind of assignment problem.

Related terms:

Variables - which need to be assigned value. (distinct & finite)

Domains - possible values (all) of a variable

Ex: V = { day, date, m, y, hour }

D = { d_1, d_2, d_3, d_4, d_5 }

d_1 → M, T, W, Th, F, S, Su.

d_2 → 1 .. 31

d_3 → Jan, Feb, ..., Dec

d_4 → 2000 .. 3000

d_5 → 1 .. 24

Constraints - single or multiple conditions which must be satisfied. (gt if the relationship between the domains)

⇒ CSP is also called domain reduction/ constraint propagation logic.

Cryptoarithmetic problem

```
  F A T
+ T H A T
---------
A P P L E
```

(Total 6 distinct alphabets) [variables]

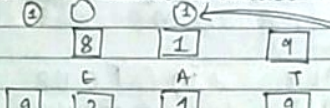
Each alphabet must be assigned a value between 0 & 9. This becomes the domain.

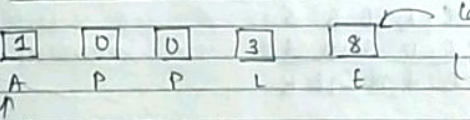
Constraints:
C_1 = Distinct no. to each alphabet
C_2 = Arithmetic operation rule must not be violated.

⇒ The domain is reduced each time an alphabet is assigned a no.
Ex - A is assigned 1
Domain now becomes 0,2,3,4,5,6,7,8,9

∴ It is called domain reduction.

⇒





May have more than one solution (cryptarithmetic)

Ex: i)

	9	5	6	7
	S	E	N	D
+	1	0	8	5
	M	O	R	E
	1	0	6	2
	M	O	N	E

ii)

	7	4	8	3
	B	A	S	E
+	7	4	5	5
	B	A	L	L
	1	4	3	8
	G	A	M	E

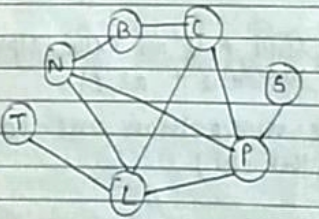
iii) GERALD + DONALD = ROBERT

iv) FOUR + FOUR = EIGHT

v) TWO + TWO = FOUR

vi) CROSS + ROADS = DANGER

Ex:



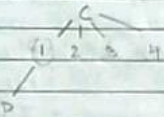
4 companies

Constraints:

i) No of students wants to appear for N, B & C
ii) No. of students want to appear for L, P & N
iii) S & P
iv) L & C
v) T, L & N
vi) C & P.

4 dates are available and a student appearing for a company cannot appear for any other company on the same date/day.

T can only come on the first date.



9/8/19

Game playing with search (Adversarial Search)

At initial level:

has some weightage w

will have the same weightage, w.

will have the same weightage, w_2

Based on these weightage, the next best move can be chosen. (heuristic func.)

The output can be :
- Player 1 wins
- Player 2 wins
- Tie

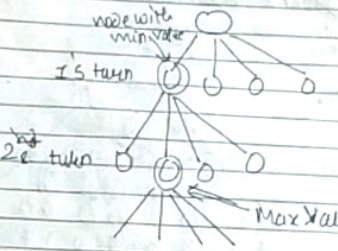
Depending on how we want the game to end, the heuristic func is selected.

Calculate weightage as
+ when move in favor of player
− when move in favor of opponent.

If both the players are using the same heuristic function. One should try to maximize it & other should minimize it

Ex: Heuristic func. - no. of balloons

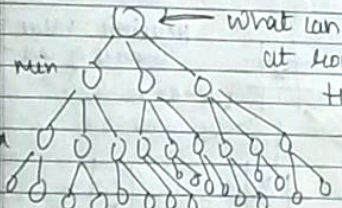
⇒ 1 : Min
2 : Max
(Tic-Tac-Toe)



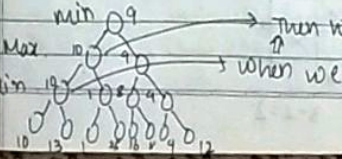
Outcome: 1 wins if: 1 (Max. value)
Tie if: 0
2 wins if: -1 (Min. value)

This is MiniMax algo. & can be applied only if the universe of the game is predictable.

→

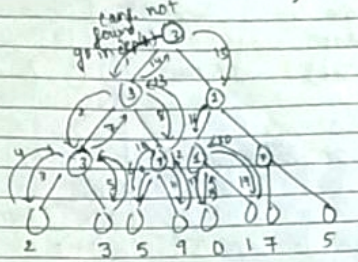


← What can be maximum value at root when we have the func. seq-as 'Minimizing' & 'Maximizing' assuming that the opponent is optimal, this searching assures the best possible way to win a game & if not, atleast end in a tie.



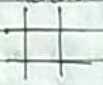
19/8/19

local search (Ex - Hill climbing)
global search (Ex - A*)



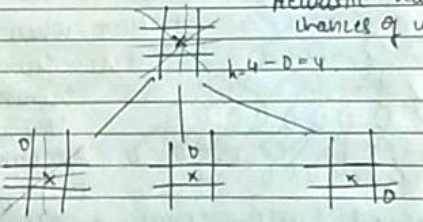
Root node cannot have a heuristic value as it cannot be defined.

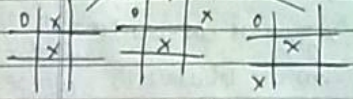
Ex:



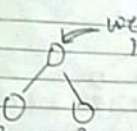
x - +1 point
0 - -1 "
tie - 0 "

True, the value for initial state cannot be defined.

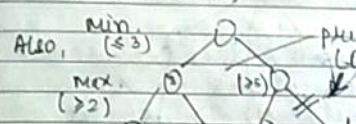




If both the players are playing optimally. What is the maximum possible value that can be achieved at the root node (the goal of MinMax algo)

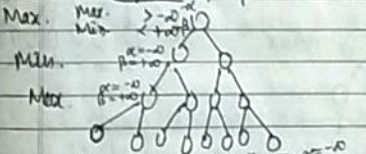


Algo,



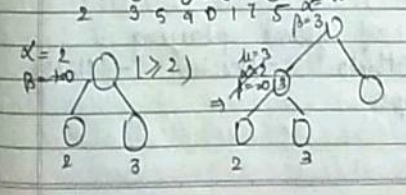
(no need to visit this node as at the upper level, we need a value ≤ 3 which is clearly not going to be satisfied because of (≥ 5)

α-β pruning



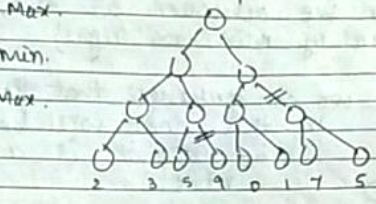
Max. Min.

Define α & β as a range of values which the node can take.

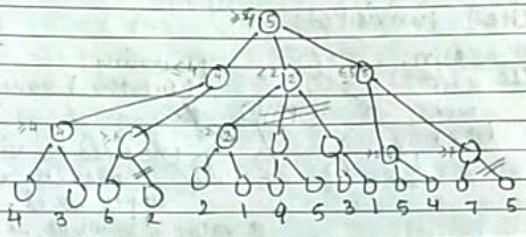


△ - max □ - max if nothing is specified
▽ - min. ○ - min root node is considered as max. node

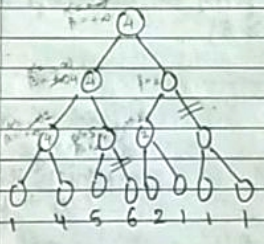
At any point, if $\alpha \geq \beta$ (if upper larger) it is time to prune the branch



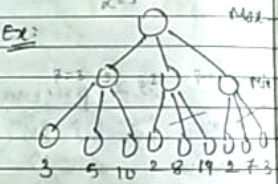
Ex:



Ex:



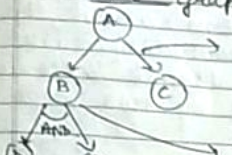
Ex:



20/8/19

Application of game playing:
- go - Othello - Checkers - Chess
- AND-OR Graph

And - Or Graph



→ can take path B Or C

→ To move further from B, both D & E steps are compulsory.

Heuristic value = h(D)+h(E)

B ^ E → C // Steps B & E need to be compulsorily performed to reach step C.

Knowledge Representation

Representation of information is important to predict results and teach a machine what task is to be done and how it can be done

Ways: i) Proposition logic
ii) Predicate logic
iii) Semantic Net
iv) Frame representation
v) Script
vi) Fuzzy logic (Uncertainty)

Proposition logic

Assign a proposition to every sentence

T ← Today is Tuesday

R ← It is raining.

No. of operations are performed to take or arrive at a decision.

These steps can be considered as a part of reasoning (Reason about the answer, if one is available). This reasoning is called logical reasoning & it can be done by two ways.

i) Forward chaining
ii) Backward chaining

Or facts driven or knowledge driven

If we are starting with initial facts it is forward chaining.
If we are starting from goal & want to find the fact for it, it is backward chaining (Prolog uses backward chaining)

FORWARD CHECKING	BACKWARD CHECKING
* Starts with facts & rules	* Starts with goal
* If no. of goal state are more	* If no of goal state are less or single.

Ex - To determine whether person belongs to Gandhi family or not
Two ways:
i) Generate entire family tree
ii) Start with the person and search for Mahatma Gandhi at every parental level. (Backward checking)

Here, Backward chaining is more feasible as the no. of nodes generated are comparatively less.

Page No.
Date

2/8/17 Knowledge Representation & Reasoning

lang → sentence → Syntax (grammar rules) / Semantic (truth value)

what else you can derive (Proof procedure / knowledge inferencing)

→ The representation varies from person to person or lang. to lang. What is true for one lang. may not be true in some other lang.
The Truth value depends on some prior knowledge

If we have some fact info. and some new inference can be made or new info derived, it is called entailment.

Any lang. should have:
- Alphabets (Variables)
- connectives ($\wedge, \vee, \rightarrow, \leftrightarrow, \neg, \oplus$)

Propositional logic
- Less expressive
- Atomic lang. - Every sentence is represented by an atomic variable

$(\neg, \vee)$ is a complete set of notations. (we can represent almost everything with these notations). Hence, we prefer writing in disjunctive form.

Proof Processing
Two ways: i) Truth table
ii) Rules

Two-valued logic - Sentence can either be True or false. (in propositional logic)

When going with proof preprocessing, the above Truth table may have three outcomes:
i) Tautology - Every case has one value - true.

ii) Contradiction - Every case has one value false

iii) Satisfiable - Some of the cases have one value true.

Ex: Your friend, F, is organizing a party
F has A, B, C, D, E friends.
Max. friends should be able to come
Conditions:
i) If B & A both appear, then E will not appear and D will not appear
ii) if B & E both appear.

ii) E will appear if C & D appear

iii) C will only appear if A also appears

iv) A will only come if also B or C appear.

→ ① $A \wedge B \rightarrow \bar{E}$ OR $A \wedge B \rightarrow \neg E$
② $B \wedge E \rightarrow \bar{D}$
③ $C \wedge D \rightarrow E$
④ $A \rightarrow C$
⑤ $B \vee C \rightarrow A$

On the end, ① ∧ ② ∧ ③ ∧ ④ ∧ ⑤

⇒

P	q	$P \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

P	q	$p \wedge q$	$p \vee q$	$\neg(p \wedge q)$	$p \vee q \wedge \neg(p \wedge q)$	$P \oplus q$
1	1	1	1	0	0	0
1	0	0	1	1	1	1
0	1	0	1	1	1	1
0	0	0	0	1	0	0

P	q	$p \rightarrow q$	$\neg p \rightarrow \neg q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p) \equiv p \leftrightarrow q$
1	1	1	1	1	1
1	0	0	0	1	0
0	1	1	1	0	0
0	0	1	1	1	1

- Puzzle: Knight and Knave puzzle.

21/8/19 $A \rightarrow B$
- if A then B
- A implies B
- B if A

- Whenever A, B
A is sufficient condition

Equivalences

$P \vee \neg P = T$

$P \wedge \neg P = F$

$P \vee F = P$

$P \wedge T = P$

$P \wedge F = F$

A unless B $\equiv \neg B \rightarrow A$

Associativity

$(p \vee q) \vee r \equiv p \vee (q \vee r)$

Distribution

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

$p \rightarrow q \equiv \neg p \vee q$

$p \leftrightarrow q = (p \wedge q) \wedge q \rightarrow p$

Example:

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

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

$\equiv T \wedge (\neg p \vee q)$

$\equiv \neg p \vee q$

(1) ⇒ 1) If John comes he will be very hostile if Sarah is there

2) Sarah will only come if Kim will be there also

3) Kim says she will not come unless John does.

⇒ $(p \wedge q \wedge r) \to r$

p	q	r	$p \wedge q \wedge r$	$(p \wedge q \wedge r) \to r$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

∴ $(p \wedge q \wedge r) \to r$ is a tautology.
(& satisfiable)

Ans (1) $S \to \bar{J}$

$K \to S$

$J \to K$

$S \to K$

$\bar{J} \to K$

S	J	K	① $S \to \bar{J}$	② $K \to S$	③ $J \to K$	④ $\bar{J} \to K$	① ∧ ② ∧ ③
0	0	0	1	1	1	1	1
0	0	1	1	0	1	1	0
0	1	0	1	1	0	1	0
0	1	1	1	0	1	1	0
1	0	0	1	1	1	1	1
1	0	1	1	1	1	0	1
1	1	0	0	1	0	1	0
1	1	1	0	1	1	1	0

J	$\bar{J}$	K	$\bar{K}$	$\bar{J} \to K$	$\bar{K} \to J$
0	1	0	1	0	1
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	0	1	1

Q A very special island is inhabitated by knight and knave only. Knights always tell truth and knave always lie. You meet two inhabitants A & B. A says that B is knave and B says that 'Neither A nor I are knaves'. Who is knight and who is knave.

→ A : A is knight
B : B is knight

A	B	¬B	A ∧ B
1	1	0	1
1	0	1	0
0	1	0	0
0	0	1	0

A is knight
which means what he says (B is knave)
must be true
It is true (B=0)
And
what B says must be false
which is also satisfied

⇒ A : I am not knave
B : A & C are knaves ¬(A∧C)
C : I and A are knight

A	B	C	¬A ∧ ¬C	A ∧ C
0	0	0	1	0
0	0	1	0	0
0	1	0	1	0

A, C knight
B knave

0	1	1	0	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	0
1	1	1	0	1

If we consider,
Socrates is a man.
All men are mortal.
We cannot represent this in eg. (propositional logic)
we need some more expressive lang.
Predicate logic is such lang.

Ex: All men are mortal
$\forall_x \; man(x) \rightarrow mortal(x)$

Some students are sleeping.
$\exists_x \; students(x) \wedge sleeping(x)$

Ex: 1) Marcus was a man — man(Marcus)
2) Marcus was a Roman — Roman(marcus)
3) All men are people — $\forall_x \; man(x) \rightarrow people(x)$
4) Caesar was a ruler — ruler(Caesar)
5) All Romans were either loyal to Caesar or hated him — $\forall_x \; roman(x) \rightarrow$ loyal-to(x, Caesar) $\vee$ hated(x, Caesar)
6) Everyone is loyal to someone — $\forall_x \; \exists_y \; loyal\text{-}to(x,y)$
7) People only try to assassinate rulers they are not loyal to — $\forall_x \; people(x) \wedge ruler(y) \wedge try\text{-}to\text{-}assassinate(x,y) \rightarrow \neg loyal\text{-}to(x,y)$

8) Marcus tried to assassinate Caesar.
Try to assassinate (Marcus, Caesar)

⇒ Was Marcus loyal to Caesar ?
loyal to (Marcus, Caesar)?
Techniques:
- Resolution
- Forward chaining
- Backward chaining

Resolution:
Steps
i) negate the question.
¬ loyal_to (Marcus, Caesar)

ii) Representation in CNF (Conjunctive Normal form).

CNF:
a) Eliminate all implications
i.e. $p \rightarrow q \equiv \neg p \vee q$

b) Reduce the scope of all negation to a single term.
i.e. $\neg(A \wedge B \wedge C) \vee D$
$\equiv \neg A \vee \neg B \vee \neg C \vee D$

c) Make all variable's name unique
$\forall_x$ man(x) then
$\forall_x \exists_y$ loyal (x, y) cannot use 'x'

$A \wedge B \Rightarrow$ i) A
ii) B

d) Move quantifiers left. $(\forall_x, \exists_x)$

e) Eliminate universal quantifiers $(\forall)$

f) Eliminate existential quantifier $(\exists)$

g) Convert to conjunction of disjunction

h) Create separate clauses for each conjuct.

Ans:

1) man (Marcus)
2) Roman (marcus)
3) $\forall_x \neg man(x) \vee people(x)$
$\neg man(x) \vee people(x)$
3) $\neg man(x)$
4) people (x)
$\Rightarrow$ ruler (Caesar)
$\forall_x \neg roman(x) \vee loyal\_to(x, caesar) \vee hater(x, caesar)$
5) $\neg roman(x)$
6) loyal-to $(x, caesar)$
7) hate $(x, caesar)$

$\forall_x \exists_y$ loyal-to (x, y)
8) $=$ loyal-to (p, q)

$\neg(\forall_x$ people $(a) \wedge$ ruler $(b) \wedge$ try-to-assassinate(a,b)
$\vee \neg$ loyal-to (a, b)
$=$ $\neg$people (a) $\vee$ $\neg$ruler (b) $\vee \neg$ try-to-assassinate (a,b)
$\vee \neg$ loyal-to (a, b)
(a) (b) (c) (d)
(e) (f) (g) (h)

Now, ¬ loyal_to(marcus, caesar) is nullified by loyal_to(p,q) where p is marcus & q is caesar (when the new fact is added in the database)

But, when asked that marcus was not loyal to caesar: the database exhausts and we know that the statement is true.

28/8/19 Reasoning

- Resolution (Proof by contradiction)
- Forward chaining (Data-driven)
- Backward chaining (goal-driven)

1. Anyone passing his KS exam and winning the lottery is happy.

$\forall_x$ pass_exam(x, KS) $\wedge$ win(x, lottery) $\Rightarrow$ happy(x)

2. Anyone who studies ~~all~~ or is lucky can pass all his exam.

$\forall_x \forall_y$ studies(x) $\vee$ lucky(x) $\Rightarrow$ pass_exam(x, y)

3. John did not study but he is lucky.

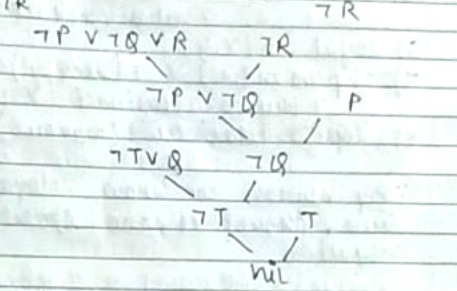
¬ studies(John) $\wedge$ lucky(John)

4. Anyone who is lucky wins the lottery.
$\forall_x \text{lucky}(x) \rightarrow \text{win}(x, \text{lottery})$

Ex: given
P
$(P \wedge Q) \rightarrow R$
$(S \vee T) \rightarrow Q$
T

CLAUSES
P
$\neg P \vee \neg Q \vee R$
$\neg S \wedge \neg T \vee Q$
$\neg S \vee Q$
$\neg \neg T \vee Q$
T

Add
$\neg R$



∴ $\neg R$ is wrong
$\Rightarrow$ R is correct
(Proof by contradiction)

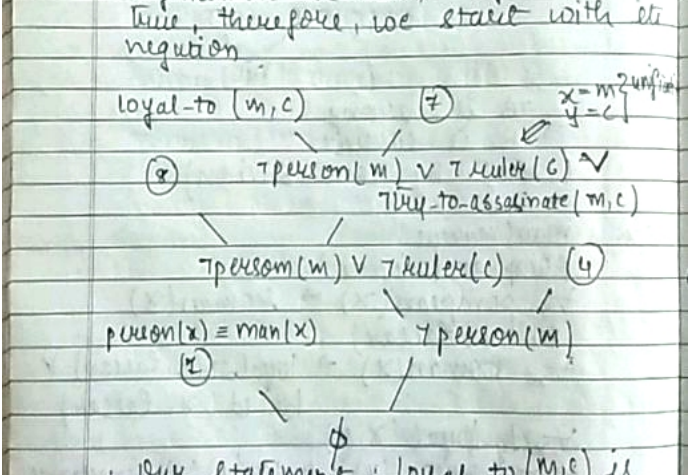
Ex: man(marcus)
pompean(marcus)
$\forall_x$ pompean(x) $\rightarrow$ roman(x)
ruler(caesar)
$\forall_x$ roman(x) $\rightarrow$ loyal-to(x, caesar) $\vee$ hate(x, caesar)
$\forall_x \exists_y$ loyal(x, y)
$\forall_x \forall_y$ person(x) $\wedge$ ruler(y) $\wedge$ try-to-assassinate(x, y) $\Rightarrow$ $\neg$ loyal-to(x, y)

Try-to-assainate (marcus, caesar)

Was marcus loyal to caesar?
loyal-to (marcus, caesar)

1) man (marcus)
2) pompean (marcus)
3) ¬pompean(x) ∨ roman(x)
4) ruler (caesar)
5) ¬roman(x) ∨ loyal-to(x, caesar) ∨ hates (x, caesar)
6) loyal-to (x, y)
7) ¬person(x) ∨ ¬ruler(y) ∨ ¬try-to-assainate (x, y) ∨ ¬loyal-to(x, y)
8) Try-to-assainate (marcus, caesar)

∵ By intution we know, ¬loyal-to(m, c) is True, therefore, we start with its negation



∴ Our statement : loyal-to (m, c) is

wrong and true. we proved that marcus was not loyal to caesar.

Ex: Prove that John is happy

happy(John)

⇒ ¬happy(John) (1)

{x/John}

(2) ¬pass-exam(J,KS) ∨ ¬win(J,lottery)

¬pass-exam(J,KS) ∨ ¬lucky(J) (3)

¬studies(J) ∨ lucky(J)

¬pass-exam(J,KS) (2)

¬studies(J) ∨ pass-exam(J,KS)

¬lucky(J) ∨ pass-exam(J,KS)

¬lucky(J)

Q. Prove hated(m,c)

Backward chaining

⇒ No need to convert in CNF.

¬loyal-to(m,c)

AND-OR Graph

All are True

person(m) — ruler(c) (True) — ¬try-to-assassinate(m,c) (True)

man(m) (True)

∴ All are true facts, ¬loyal-to(m,c) is correct.

Ex: Law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Col. West who is American.

Prove that Col West is criminal.

→ American (x) ∧ weapon (y) ∧ sells (x, y, z) ∧ hostile (z) → criminal (x)

$\exists_x$ Owns (Nono, M_1) ∧ Missile (x)

$\forall_x$ Missile (x) ∧ Owns (Nono, x) → Sells (West, x, Nono)

Missile (x) → Weapon (x)

Enemy (x, America) → Hostile (x)

American (West)

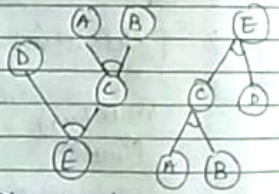
Enemy (Nono, America)

29/8/19

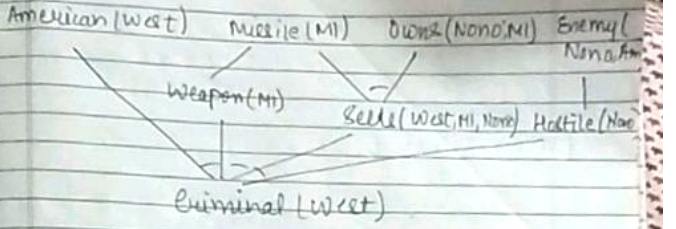
KB: $A \wedge B \Rightarrow C$
$C \wedge D \Rightarrow E$
$C \wedge F \Rightarrow G$

Facts: F1 : A
F2 : B
F3 : D

E ?



Forward chaining
Col West is criminal proof



⇒ FC is data-driven, automatic, unconscious processing
BC is goal driven, appropriate for problem solving & has lesser complexity

→ $\exists x$ Dog(x) ∧ Owns(Jack, x)
$\forall x$ [$\exists y$ Dog(y) ∧ Owns(x, y)] → AnimalLover(x)
$\forall x$ AnimalLover(x) → ($\forall y$ Animal(y) → Kills(y)
Kills(Jack, Tuna) ∨ Kills(curiosity, Tuna)
Cat(Tuna)
$\forall x$ Cat(x) → Animal(x)