Distinguishing characteristics of GAs

→ Does not require any auxiliary information

Appln areas by domain

AI Production System: 16/9/19

shows a separation betⁿ data, operation, control

↓

select a path among a set of alternatives

rule selection

control strategy

- irrevocable
- tentative
  - backtracking
  - graph search control

Algo.

① Procedure Production

Step1 : Data is initial database

Step2 : Until data satisfies the termination condition to begin, select some rule R in the set of rules, that can be applied to data

Step3 : Data

First Order

Syntax:

Components

Eg:

Scanned by CamScanner

Step3: Data is the result of applying R to data.

possible: top()
left()
right()

First Order Predicate Logic (FOPL) → does not allow quantification over predicates / functions.

syntax: → alphabet of symbols & how wff is formed (well-formed formula)

Components: predicate symbol
var "
function "
constant "

Eg: Nielson wrote Principles of A.I.

predicate ⇓ represents a relation

Represented as: write (Nielson, Principles of AI)
⟹ TRUE

To get who wrote it?
write (X, Principles of AI)

write (x, y) ⇒ who wrote something?
→ Inf. searches needed.

→ John's brother & John's sister are sibling to each other. brother (x, John)
sibling (John's brother, John's sister)
→ sister (x, John)

Scanned by CamScanner

OR sibling (brother (John), sister (John))

predicate → return T/F

func^n → return value

→ The house is yellow.

yellow (h1) OR color (h1, yellow)

predicate

CONNECTIVES :

→ John lives in a yellow house

color (h1, yellow) ∧ live (h1, John)

→ John plays chess or badminton

play (John, chess) ∨ play (John, badminton)

→ If the car belongs to John, then it is grey

belong (c1, John) ⇒ color (c1, grey)

→ All elephants are grey

∀(x) elephant (x) ⇒ color (x, grey)

→ There is a person who wrote P.A.I

∃(p) person (p) ⇒ wrote (P, PAI)

PL < FOPL < SOPL

2nd order P.L.

Scanned by CamScanner

$\neg (\exists(x)\ P(x)) \Rightarrow \forall(x) \neg P(x)$

→ For every set x, there is a set y such that cardinality of y is greater than that of x

$\forall(x) \left(\exists(y) \left(\text{set}(x) \wedge \text{set}(y) \Rightarrow (\text{card}(y) > \text{card}(x)) \right) \right)$

~~$\forall(x) (\text{card}(x,x) \wedge \exists(y))$~~

$\forall(x) \left(\text{set}(x) \Rightarrow \exists(y) \exists(m) \exists(n) \left(\text{set}(y) \wedge \text{card}(y, n) \wedge \text{card}(x, m) \wedge \text{greater}(n, m) \right) \right)$

23/10/19

Most General Unifier

$$(\forall x)\ \{P(x) \Rightarrow \{(\forall y)[P(y) \Rightarrow P(f(x,y))]$$
$$\wedge \sim (\forall y)[Q(x,y) \Rightarrow P(y)]\}$$

Steps

1. Eliminate $\Rightarrow$
2. Reduce scope of negation
3. Standardize variables
4. Eliminate $\exists$
5. Convert to Prenex
6. Put matrix in CNF
7. Eliminate $\forall$
8. Separate into clauses (Replace $x_1 \wedge x_2$ with set of wff (well formed formula): $\{x_1, x_2\}$)
9. Rename Var

---

1. $A \Rightarrow B$ to $\sim A \vee B$

$$\forall x \{\sim P(x) \vee (\forall y)[\sim P(y) \vee P(f(x,y))]$$
$$\wedge \sim (\forall y)[\sim Q(x,y) \vee P(y)]\}$$

Note:

(2)

$$(\forall x)\{\sim P(x) \vee (\forall y)[\sim P(y) \vee P(f(x,y))] \wedge (\exists y)[Q(x,y) \wedge \sim P(y)]\}$$

After applying NOT, $\forall$ changes to $\exists$ & then apply De Morgan's Law

(3) dummy variable ⇒ same variable name should not be used anywhere else

$$(\forall x)\{\sim P(x) \vee (\forall y)[\sim P(y) \vee P(f(x,y))] \wedge (\exists w)[Q(x,w) \wedge \sim P(w)]\}$$

Eg: $\forall x[P(x) \Rightarrow (\exists x)Q(x)]$

$\downarrow$

$\forall x[P(x) \Rightarrow (\exists w)Q(w)]$

(4)

Eg: $(\forall y)[(\exists x)P(x,y)]$

i.e. there exists an x s.t. x & y are related

So write $x = f(y)$

$\Downarrow$

$(\forall y)[P(f(y),y)]$

$$(\forall x)\{\sim P(x) \vee (\forall y)[\sim P(y) \vee P(f(x,y))] \wedge [Q(x,h(x)) \wedge \sim P(h(x))]\}$$

Note: If $(\exists x)$ is not prefixed with $(\forall y)$ ⇒ So x is represented by a const operator (represented in CAPITAL)

Scanned by CamScanner

ie. $(\exists x)\, P(x) \rightarrow P(H)$

Skolen constant

⇓

Process is called skolenization

(5) Prenex ⇒ all $\forall$ quantifiers are at the front of the expression.

Only quantifier part (Prefix), Quantifier free part (matrix)

$$(\forall x)(\forall y)\{\underbrace{\sim P(x)}_{A} \vee \{\underbrace{[\sim P(y) \vee P(f(x,y))]}_{B} \wedge [Q(x, h(x)) \wedge \sim P(h(x))]\}\}$$

(6) $x_1 \vee x_2 \wedge x_3 = (x_1 \vee x_2)(x_1 \vee x_3)$ ⇒ Conjunctive Normal form

~~$(\forall x)(\forall y)\{\sim P(x) \vee \sim P(y) \vee P(f(x,y))\}$~~

(7)

$A + B.C = (A+B)(A+C)$

(8)

$(\forall x)(\forall y)$

$\{[\sim P(x) \vee \sim P(y) \vee P(f(x,y))]$

set of wff : $\{$

Scanned by CamScanner

$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$$

$$(\forall x)(\forall y)\{\sim P(x) \vee [\sim P(y) \vee P(f(x,y))] \wedge [\sim P(x) \vee [Q(x,h(x)) \wedge \sim P(h(x))]]$$

(A) (B) (C)

$$\wedge [\sim P(x) \vee Q(x,h(x))] \wedge [\sim P(x) \vee \sim P(h(x))]$$

(7) Just remove $\forall$

(8) Clauses: (those with only $\vee$ & are separated by $\wedge$)

set of wff: $\{(\sim P(x) \vee \sim P(y) \vee P(f(x,y))), (\sim P(x) \vee Q(x,h(x))), (\sim P(x) \vee \sim P(h(x)))\}$

Scanned by CamScanner

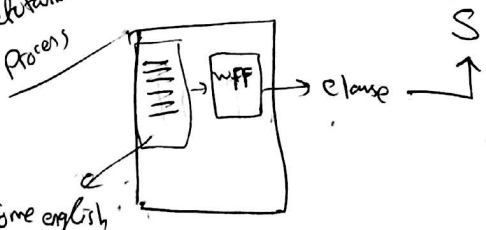
(9) $\{ \sim P(x_1) \vee \sim P(y) \vee P(f(x_1, y)),$

$\sim P(x_2) \vee Q(x_2, h(x_2)),$

$\sim P(x_3) \vee P(h(x_3)) \}$

Parent Clauses	Resolvent	Comments
P and $\sim P \vee Q$	Q	Modus Ponens
$P \vee Q$ and $\sim P \vee Q$	Q	Merge
$P \vee Q$ and $\sim P \vee \sim Q$	$(P \vee \sim P$ and $Q \vee \sim Q) = 1$	Tautology
$\sim P$ and P	NIL	Empty Clause → sign of contradiction
$\sim P \vee Q$ and $\sim Q \vee R$	$P \Rightarrow R$ ($\because P \Rightarrow Q$ & $Q \Rightarrow R$)	Method due to chaining (Transitive)

Resolution Refutation Process



Task is to find out whether W logically follows from S.

Prod

Let S

Procedure

Step 1

1
2
3

Eg:

1
2
3

Pro

Production System for Resolution Refutation

Let S is a set of clauses (best-set)

Procedure Resolution

Step 1 ① Clauses (→ S) ~~NIL is a~~

② until NIL is a member of clauses do

③ begin

④ select two distinct resolvable clauses C_i & C_j in Clauses (from Clause-Set)

⑤ Compute a Resolvent (R_{ij}) of C_i & C_j.

⑥ Clauses → the set produced by adding R_{ij} to Clauses.

⑦ end

Eg: ① Whoever can read, is literate

② Dolphins are not literate

③ Some dolphins are intelligent

Prove that some who are intelligent cannot read.

Scanned by CamScanner

A

S
① $(\forall x)$ Read $(x) \Rightarrow$ Literate (x) — where
② $(\forall y)$ Dolphin $(y) \Rightarrow \sim$ Literate (y)
③ $(\exists z)$ Dolphin $(z) \wedge$ Intelligent (z)
④

$W = (\exists w)$ Intelligent $(w) \wedge \sim$ Read (w)

Break to clause form

$S \cup \sim W$
$= S \cup W$

① $\sim R(x) \vee L(x)$
② $\sim D(y) \vee \sim L(y)$
③ $D(C) \wedge I(C)$

since separated by AND so 2 clauses

there exist, but NO $\forall$. So replace variable by a const C

(∵ we need to refute)

$\overline{W} = (\forall w) \sim I(w) \vee R(w)$

So clauses → $\sim I(w) \vee R(w)$

So we get 5 clauses.

$S = \{ \sim R$ ... $\}$

Tree

$\sim R(x) \vee L($

$\sim R$

Scanned by CamScanner

> Literate (x)

) ⇒ ~Literate (y)

∧ Intelligent (z)

) ∧ ~ Read (ω)

S ∪ ~W
= S ∪ W

there exist, but NO ∀. So replace variable by a const C)

R (ω)

υ] ∨ R (ω)

$$S = \begin{cases} \sim R(x) \vee L(x), \\ \sim D(y) \vee \sim L(y), \\ D(c), \\ I(c), \\ \sim I(\omega) \vee R(\omega) \end{cases}$$

→ give separate variable for diff clause

multiple clause arising from same. give same name

· A variable can be replaced by
① another variable,
② const - symbol
③ func^n symbol

Tree

~R(x) ∨ L(x) | ~D(y) ∨ ~L(y) | D(c) | I(c) | ~I(ω) ∨ R(ω)

{x/y}

~R(x) ∨ ~D(x) | {c/ω} R(c)

{c/x}

~R(c)

NIL

{ —/— } To write beside lines.

↓ replaced by | ↓ which was replaced

variable → small letter

const → Capital letter

Scanned by CamScanner

Control Strategy for Resolution Refutation

8/11/19

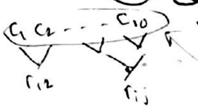
① **Breadth First**

Compute all 1st level resolvent ; then level 2, level 3 .... — Complete, inefficient

(whether providing goal wff if it exists)

② **Linear input form**

(★) Each resolvent has at least 1 parent belonging to the base set — Not Complete



(base set: initial set of clauses form the base set)

③ **Set of Support**

At least 1 parent of each resolvent is selected from among the clauses, resulting from negation of goal wffs or from their descendants — Complete

④ **Unit Preference**

try to select a single literal clause (unit) to be a parent

⑤ **Ancestry filtered from**

each resolvent has a parent that is either in base set or ancestor of other parent

⑥ **Combination**

Scanned by CamScanner

6/11/19

tion

—

uletion Providing goal wff if it exists

resolvent ;

— Complete, inefficient

least 1 parent

set — Not Complete

of
n form the
ive set

h resolvent is

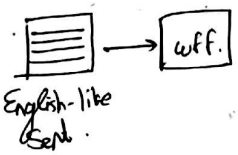
clauses, resulting
ffs or from
complete

literal clause

nt that is
estor of other

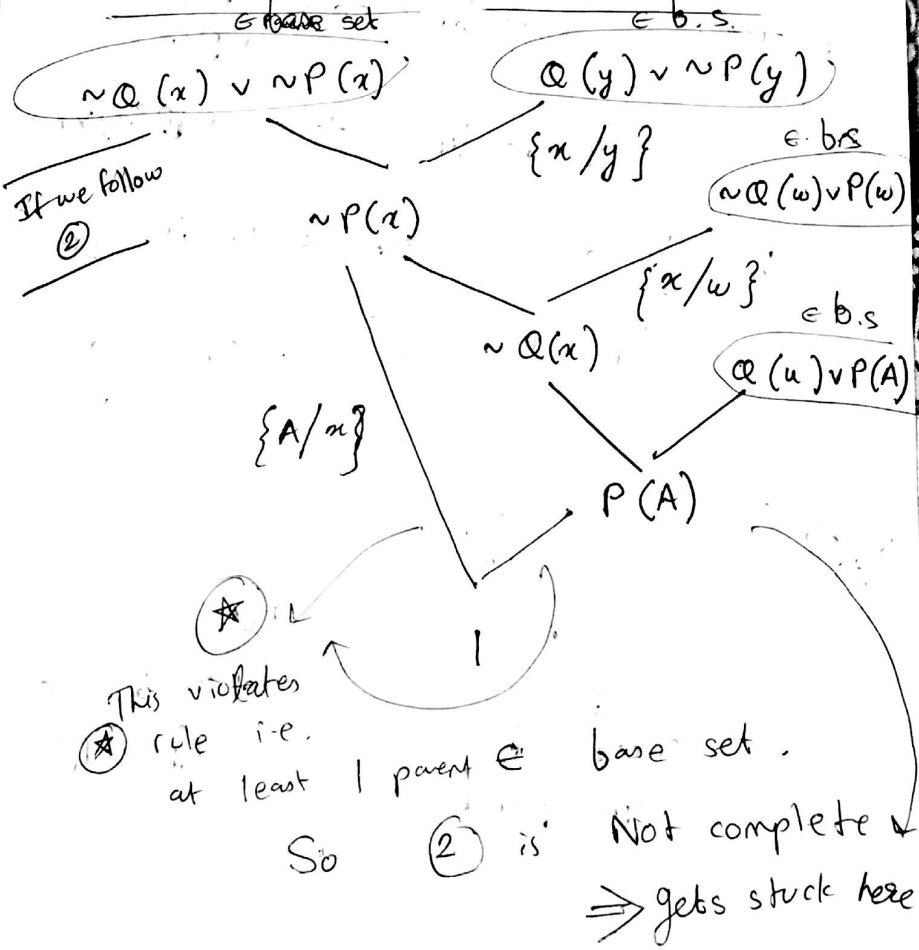
$\sim Q(x) \vee \sim P(x)$ $\quad Q(y) \vee \sim P(y)$

$\sim Q(w) \vee P(w)$ $\quad Q(u) \vee P(A)$



apply rules of inference (9 steps) & get set of clauses.

derived wff / goal wff



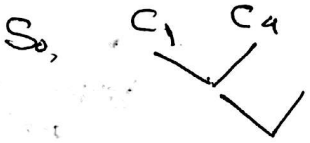
Scanned by CamScanner

If we follow (8)

See eg of dolphin

$\equiv \mid C_1, C_2, C_3$

(=) goal aff $\xrightarrow{\text{negation}}$ $= \mid C_4, C_5$.



If we follow (9)

Single literal clause : $\rightarrow P(A)$

$\rightarrow P(x)$

Double literal clause $\rightarrow \neg P(x) \vee P(x)$

We try to select single literal clause

(may not be present)

↓

so no. of steps will be reduced.

If we follow (5)

$\sim P(x)$

$P(A)$

↙ ancestor of other parent.

can be done (prev. pg)

Eg: If

Jo

A.

when we use reso... these sho... a g...

Clause

Resol...

By

u

Eg: If Fido goes wherever John goes & if John is at school, where is Fido?

OR use AT instead of P

A.

① $\forall(x), P(J, x) \Rightarrow P(F, x)$

② $P(J, \text{school})$ → const.

③ $(\exists x)\, P(F, x)$

when we use variable, there should be a quantifier

Clause forms

① $\sim P(J, x) \vee P(F, x)$ → chng to new variable y (to avoid clash)

② $P(J, \text{school})$

③ Negating → $\sim [(\exists x)\, P(F, x)]$

$\Downarrow$

$(\forall x) \sim P(F, x)$

Resolution Refutation Tree

$\sim P(F, x)$ $\qquad$ $\sim P(J, y) \vee P(F, y)$

$\{y/x\}$ $\qquad$ $P(J, \text{school})$

$\sim P(J, y)$

$\{\text{school}/y\}$

NIL → root node is NIL

By resolution refutation, we do, $S \cup \sim W$

Scanned by CamScanner

How to get answer?

→ Append to each clause arising from negation of goal wff, its own negation

$\sim P(F,x) \cup P(F,x)$ (own neg)

→ Following the str of refutation tree, perform the same resolution as before until some clause is obtained at the root.

So,

This is added now

$\sim P(F,x) \cup P(F,x)$ $\quad \sim P(J,y) \vee P(F,y)$

MODIFIED PROOF TREE

$\sim P(J,y) \vee P(F,y)$ $\quad P(J, school)$

Ans: of the given ques

$P(F, School)$

→ Use the clause at the root as an answer statement

AND-OR graph

all the nodes need to be procesed

→ hypergraph
→ hyperarc
→ K. connector
→ Formulatin
→ 2 perso
pla
→ Iterative
→ Adversar
→ Static
→ f(x

• Root → d
t
• my
• el
• Arc
• In

Scanned by CamScanner

Game Playing Strategy

6/11/19 (II)

AND-OR graph

- all the nodes need to be processed
- One of the ....

Eg. In ludo chance factor involved
You go on playing after consecutive sixes.

→ hypergraph
→ hyper arc
→ K. connector
→ Formulating game playing as search
→ 2 person, perfect info
players move alterrately, no chance factor
→ Iterative methods
→ Adversary → adverserial phenomenon takes place for playing such games - Like A moves with a plan B hv a complete diff plan to win.
→ Static Evaluation func^n

→ $f(x)$ = large +ve (my winning cond^n)
= large -ve (my losing cond^n)
= $+\infty$
= $-\infty$,
= 0 (draw cond^n)

Evaluation func^n of Tic-Tac-Toe

$f(x)$ =

weighted features.

Game Tree

- Root → decision on what is the best single move to make next
  - myturn → MAX
  - else → MIN
- Arcs - possible legal moves.
- In each level - MAX/MIN

Scanned by CamScanner

- Nodes corresponding to the MIN's next move have successors that are like — nodes (AND) (all the nodes)
- Nodes ---- to MAX's = '---- OR ('any one of the nodes) ---like — nodes

Game - #nodes in complete game tree

checkers : 10^{90}

chess : 10^{120}

↓ bcos hv to ensure that win is mine from all possible nodes. & thus make decision

(9 possible)

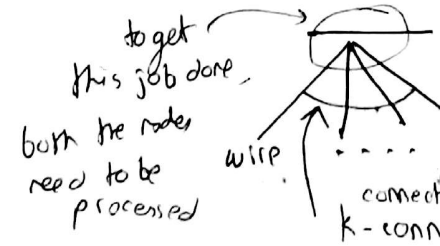
OR-graph

from this pt

I can follow only 1 path.

So only one of the nodes need to be processed

Eg: If I have a wire and a screwdriver, I could install the light.



AND-graph

Hypergraph

So, o
graph

Eg:
I
e

So,

Again,

Can
AN

A.

MIN's next move

like — nodes (all the nodes)
(AND)

= '- - -

like — nodes (any one of the nodes)
(OR)

game tree

↙

bcos hv to ensure that win is mine from all possible nodes.
& thus take decision

(9 possible)

— from this pt

— I can follow only 1 path.

of the nodes need to be processed

a screwdriver,

& the light)

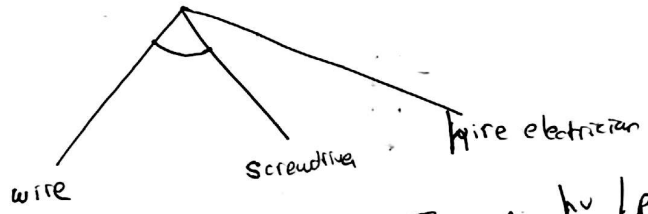
AND-graph

/Hypergraph

So, OR-graph can be differentiated from AND-graph by the help of a connector.

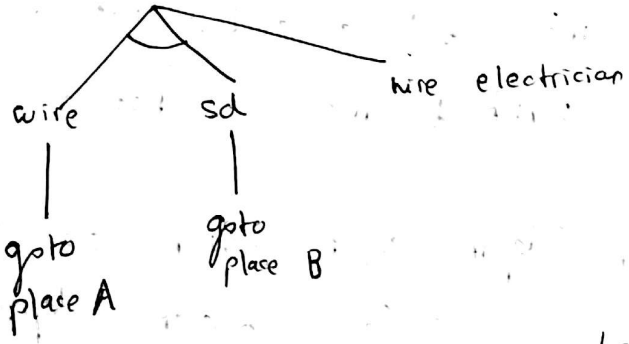
Eg: If I have a wire & a screwdriver, I could install the light else, hire an electrician.

So,



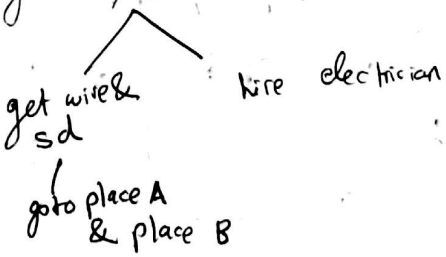
So, in 1 single graph, I can hv 1 part as AND & one as OR

Again, this can be extended to



Can this be converted to a graph w/o AND? (using only OR)

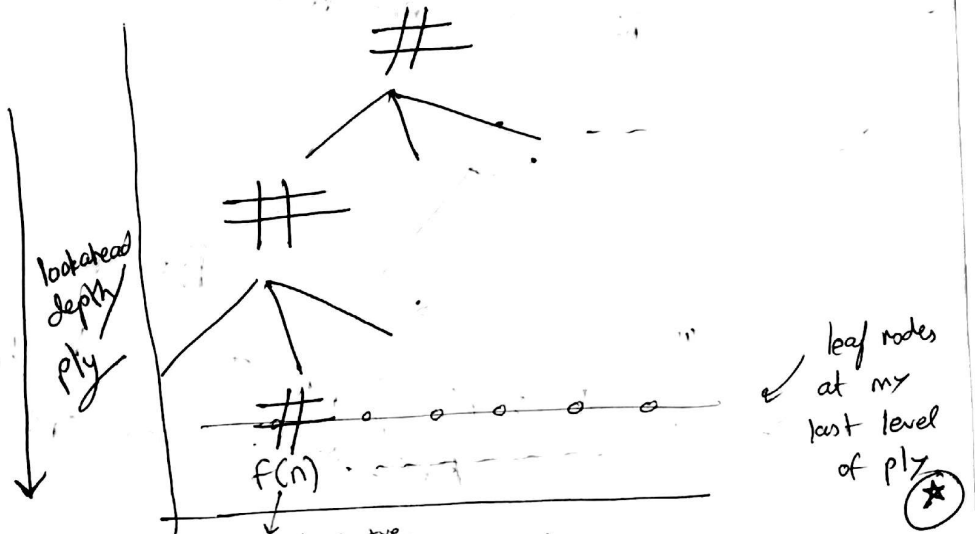
A.



Scanned by CamScanner

This concept can be used for formulating a game where 2 persons are involved.

In chess 10^{90} nodes in a complete tree → million yrs to compute.



Always we try to find out all possible moves till a certain extent to decide which to move (Not only the current move)

The depth upto which one could foresee is called lookahead depth / ply

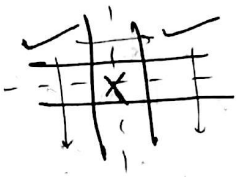
---

(*) For each of these so called leaf nodes Some static evaluation func^n performed to conclude how good that move is wrt root node.

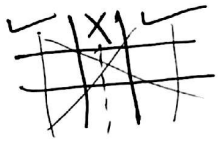
Scanned by CamScanner

Eg: For tictactoe, Static eval func^n (may be) determined by no. of open rows/col/diag to form 3-consecutive (no. of 3-lengths open for me)

Eg:



✓ ⇒ 3
X ⇒ 2



✓ ⇒ 4
X ⇒ 1

So, one possibility, static eval func^n

$$= \left(\begin{array}{l}\text{No. of 3 len open for me} \\ \text{minus} \\ \text{No. of 3 len open for opponent}\end{array}\right)$$

For chess

$f(x)$ = weighted features

$= w_1 f_1 + w_2 f_2 + \cdots$

↓

can be weight of pieces AND can also include its pos^n on board

leaf nodes at my last level of ply ★

possible

decide

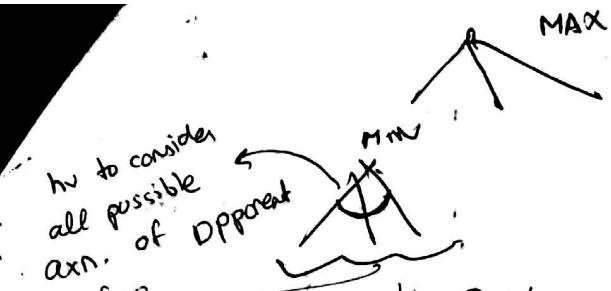
current move)

foresee

ply

leaf nodes performed w.r.t root node

Scanned by CamScanner



From MAX's pt of view my next move can be considered by any one of the node

From MIN's POV, my next move hv to be determined from all these nodes (So that I win from all these nodes)

→ At

(5) Pic ch de

Eg:

Oppo's m

D

100

(start from here, go up)

the

F

So ch a

Searching GameTree using Minimax Algo

Step (1): Create start node as a max node (my turn to move) with current board configuration

(2) Expand nodes down to some depth (ply) of lookahead in the game

(3) Apply evaluation func^{n} at each of the leaf nodes

(4) Back up values for each of the non-leaf nodes until a value is computed for the root.

How to compute?

→ At MIN node, backed up value is the (min) of the values associated with its children

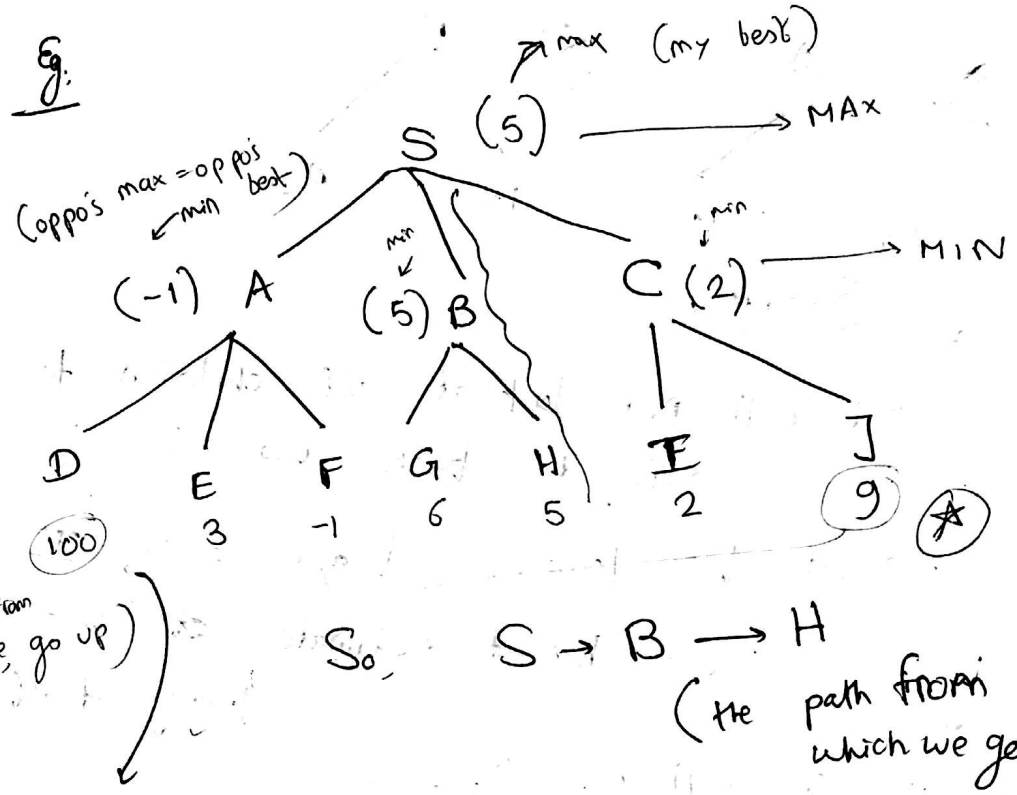
(min) ← diff bet^{n} winning chance ↓

Scanned by CamScanner

→ At MAX node, — — — max of the values ---

(5) Pick the operator associated with the child node, whose backed up value determined the value at the root.

Eg:



So, S → B → H

(the path from which we get 5 at S)

the best winning config

But my algo didn't choose this, bcos here in (-1) → when oppo gives his best, I lose.

But in S → B → G, I win irrespective of opponent config

(*) Not chosen 9 also

2 is ≃ 0. So as we go deep, its possible that we lose when we go more down.

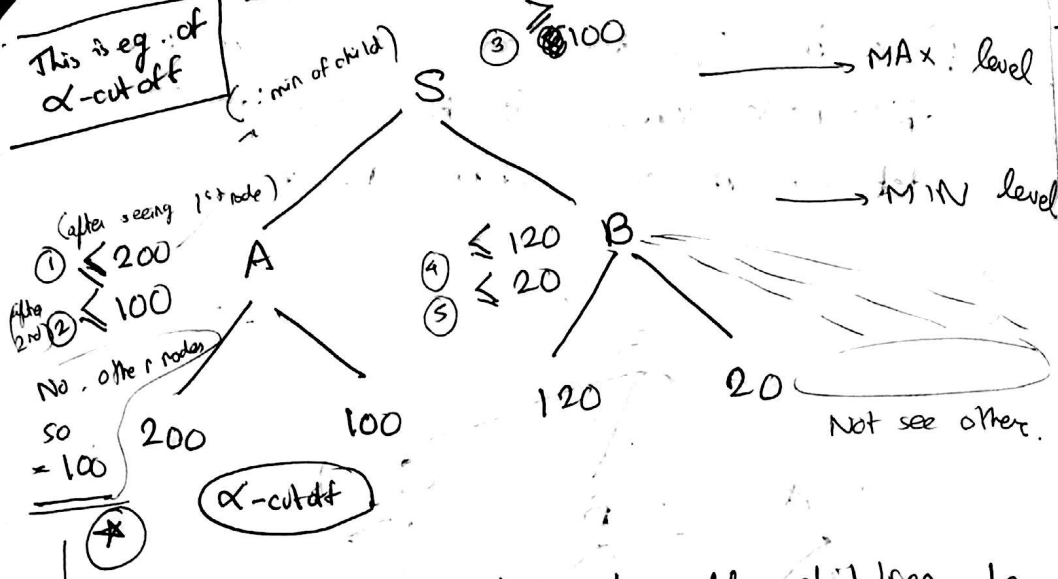
So choose such a place which can fetch better scores altogether.

Scanned by CamScanner

Alpha - Beta Pruning Algorithm:

This is eg. of α-cutoff

Eg. of β-cutoff



see Game playing

→ I will not look at all children to calculate backed up value

→ But by looking 1 after another, I can hv at estimate w/o looking at all

Now I will not look at B.

Now after ★, S ≥ 100 (∵ MAX)

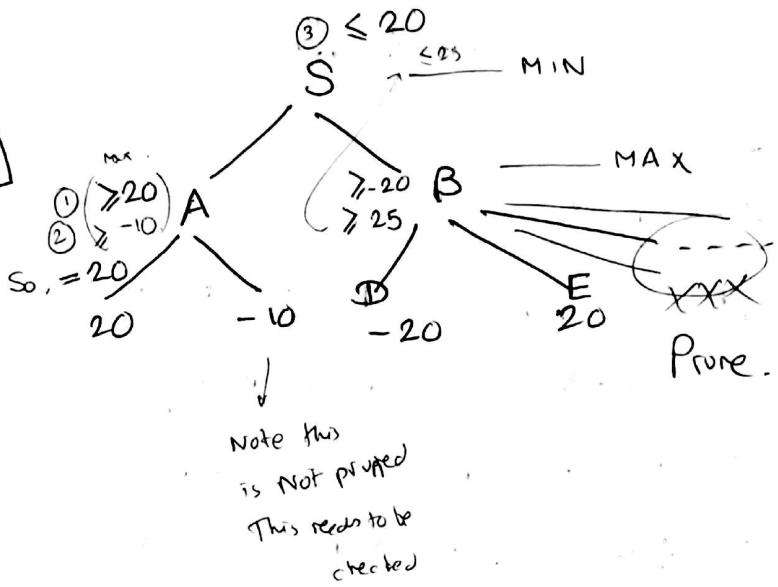
After ③ & ④, at S, its ≥ 100 & ≥ 120

After ⑤, no matter how many nodes we see, it'll always be ≤ 20 which has no effect at S (when becomes ≥ 20)

So I can prune other children at B

Scanned by CamScanner

Eg. of β-cutoff



see Tic tac toe playing in pdf α-cutoff β-cutoff

In chess branching factor = 35

After α-β pruning, it reduces to 6

Worst case : No pruning

Best case : $(2b)^{d/2}$ when is it best case?

Horizon effect

If I see till depth d

If I see till depth d+1, situation is entirely different

Uncertainty Handling

7/11/19

Reasoning based on PL

→ Reasoning based on PL operates under:

Certain, complete, unchanging, consistent facts

→ Given facts, rules → confident conclusions

→ Derived truth → never contradicts, given that no contradictions exist on axioms.

→ Real life → imprecise, incomplete, uncertain

Limitations of PL

→ limited in expressive power (more or less ---)

→ Only the True/False Statements

→ No way to produce new knowledge about the world; only what is derivable.

Uncertain Situation

→ Uncertain knowledge - Heuristic knowledge - certain sets of evidences probably imply certain conclusions

→ Uncertain data - domain knowledge is certain, but not data - to infer a specific cause from an observed effect, we may hv to rely on questionable test result.

→ Incomplete
proces

→ Random

→ PL is

NMR

→ Hu

absol
chan

truth value will not depend on other sent. & truth info never chng over a time.

→ A

→ N

Scanned by CamScanner

7/11/19

→ Incomplete info — decisions are taken in course of processing incrementally acquired info.

→ Randomness — knowledge & info is complete, data is certain but domain is inherently random - stochastic domain

→ PL is monotonic (# facts known to be T at any time)

NMR systems

→ Human problem solvers often augment absolute truths with beliefs that are subject to change given further info; Tentative beliefs
(absolute truths: truth value will not depend on other sent. & truth info never chng over a time.
beliefs: truth info may chng over a lifespan of time)

- based on default assumptions that are made in the light of lack of evidence to the contrary

→ An NMRS tracks a set of beliefs & revises those beliefs when new knowledge is observed/derived

→ NMRS - a set of premises (immutably T), tentative beliefs (potentially incorrect), dependency record for each tentative belief (tracks beliefs v/s justifications)

→ to store a belief stmt. & its set of justifications (dependency record)

Scanned by CamScanner

NMR system

→ In NMR, the new facts become known which contradicted & invalidated old knowledge → old knowledge was retracted causing other dependent know... to become invalid → thereby requiring further retractions → shrinkage or non-monotonic growth at times

→ Methods to accurately represent and deal with different forms of uncertainty ---→

4 diff approaches

(1) TMS (truth maintenance systems) — permits addn of changing (even contradictory) stmt to KB (knowledge base)

(2) Modal & temporal logic — permit representation & reasoning about necessary & possible situations, temporal & other related situations

(can get idea in Patterson book)

(3) Fuzzy logic — deals with vague & imprecise info

(4) Reasoning based on Probability

→ Since a
it i
fact
truth

→ So, u
any
indi

→ B
an
de

→ N

→ M

→ D

→ Ma

&

Scanned by CamScanner

→ Since an NMRS include tentative knowledge, it is possible to add new piece of knowledge that will cause a previously believed tentative truth to become F.

→ So, when a belief in an NMRS is revised, then any beliefs that rest on it, directly/indirectly, must also be revised.

→ Brief revision – propagates the effect of any change in belief; through dependency directed backtracking; when any inconsistency arises, as a result of newly added justification, dependency directed backtracking is invoked to restore consistency

→ NMRS → useful for planning & design (requiring large no of tentative assumptions)

– NMRS → consumes memory, time

TMS

→ Doyle, 1979

→ Main job → maintain consistency in knowledge Base (KB) being used by any prob solver & not to perform any inference fns; As such it frees the problem solver from any concerns of consistency

Scanned by CamScanner

TMS...

IE

KB

→ TMS maintains currently active belief set. Updating is incremental; after each inference info. is exchanged between IE & TMS

IE tells TMS what deductions it has made. TMS asks questions about current beliefs & reasons for failure. It maintains a consistent set of beliefs for IE

→ TMS operates as KBMS & is called every time the reasoning system generates a new Truth value. TMS, using belief revision, takes any action required to modify dependent beliefs to maintain consistency.

TMS

In TMS,

Node → one unit of knowledge: fact/rule/assertion; at any point of time, every node is in one of two conditions

IN → currently believed to be T

OUT → " " " " F

(There is no possible cond^n that would make

it T/

→ Ass
node's
e
→ F
w

Structure →

→

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

Scanned by CamScanner

it T / the conditions required to note it T are currently IN)

(support list)

SL justifications

→ Associated with each node → justif^n/ for node's truth value; assume, 1 just^n for each node

→ For each node, that is IN, TMS records a well founded support → proof of the validity of the node

Structure → [SL (in-nodes) (out-nodes)]

→ In-nodes → list of all nodes that must be IN for this node to be T

→ Out-nodes → OUT ---- T

for ① to be true ↓ should be IN (true) ↓ should be OUT (false)

SL (2) (3)

① It is sunny 1

② It is daytime 2 — SL (1) ( )

③ It is raining 3 — SL ( ) (1)

④ It is warm 4 — SL (1) (3)

these stmt have no dependency on ②

So ② is an absolute stmt.

e belief set.
each inference
=E & TMS,
it has node
beliefs &
a consistent

n called
generates a
belief
required to
tain consistency.

dge : fact/
time, every
itions
be T
" F
hat would note

Scanned by CamScanner

SL justifn

→ Premise : always valid ; in-list, out-list always empty

→ Normal deduction : It is formed in normal sense of a monotonic system ; out-list always empty

→ Assumption : a belief supported by lack of contrary info. Out-list never empty

CP Justification

→ Used to support hypothetical reasoning

→ [CP <consequent> (in-hypothesis)]

→ CP also includes out-hypothesis, always empty

→ A CP justification is valid iff the consequent node is IN whenever all of the nodes on in-hypothesis are IN.

→ Prim
is
→ Hum
w/o
→
→
Purpose
Def

Neuro Computing : an Intro

→ Primary task of all biological neural systems is to control various functions (mainly behavioural)

→ Human being can do it almost instantaneously & w/o much effort eg recognizing a scene/music immediately

→ Artificial Neural Network (ANN):
or Neural Network (NN) models try to simulate the biological neural n/w with electronic circuitry

→ Also known as Connectionists Model / Parallel Distributed Processing (PDP)

Purpose: To achieve human like performance (particularly in pattern recognition & image Processing)

Definition

Def^{n}: Massively parallel interconnected network of simple processing elements which are intended to interact with the objects of the real world in the same way as biological systems do.

→ NN models are extreme simplifications of human neural systems

→ Computational elements (neurons / nodes / processors) are analogous to the.

Similarity bet^n (Biological) BNN & ANN

→ Gets i/p via synaptic connection.

→ Accumulated i/p is transferred to a single o/p

→ O/p is transmitted through axon

→ firing of neuron

An artificial neuron

diag.

Artif. / electronic neuron

→ Gets i/p through resistors

→ Total i/p is converted to a single o/p by OP-AMP

→ --- transmitted via resistors.

Neural network

Scanned by CamScanner

ene simplifications of

eurons / nodes / processors)

& ANN

connection.

formed to a single o/p

ugh / axon

ron

a single o/p

by OP-AMP

ors.

Summary

→ An electronic neuron emulates a biological neuron

→ Artf. neurons are then connected to form a n/w to mimic (!) the topology of human nervous system

→ Func^{n} performed by a NN is determined by the n/w topology, conn^{n} strength, processing performed at computing elements / nodes, & status updating rule

General framework of neural n/w

Processing units

→ Receives i/p from connected neurons, compute an o/p value & sends it to other connected neurons

→ Three types of units – i/p, o/p, hidden

O/p value

$$O_i(t) = f(I_i(t))$$

→ Total I/p for i^{th} neuron is I_i

→ f is a threshold or squashing func^{n}.

Scanned by CamScanner

→ Unidirectional connections (w_{ij})

→ $w_{ij} < 0$ → unit u_j inhibits unit u_i

→ $w_{ij} = 0$ → unit u_j has no direct effect on unit u_i

→ $w_{ij} > 0$ → unit u_j excites unit u_i

Characteristics of neural networks

Train → supervised learning

Learn from example : shown a set of i/ps, they self-adjust to produce consistent response

Test

Generalize from prev eg to new ones : once trained a n/w's response is mostly insensitive to variations in i/p

Abstract essential chars from i/p : find the ideal (prototype) from imperfect i/ps.

→ unsupervised learning

Major advantages

adaptivity - adjusting the connection strengths to new data/information

speed - due to massively parallel architectures

robustness - to noising, confusing, ill-defined/noisy data

Scanned by CamScanner

)

ibits unit u_i

o direct effect
: on unit u_i

ites unit u_j

---

wards

---

of i/ps, they

oduce consistent
response

once trained,

tive to variations
in i/p

: find the ideals

---

n strengths to
formation

architectures

ill-defined/
noisy data

ruggedness — to failure of components

optimality — as regards error rates in performance.

Learning (parameter updating)

Associative (supervised) learning

Learning pattern pair association

Input = $X = \{x_1, \ldots x_n\}$

Output = $T = \{t_1, t_2 \ldots t_n\}$

Learn (X, T)

→ auto-associator $(T \cong X)$

→ hetero-associator (any arbitrary combination of X, T)

(classification)

Regularity detection (unsupervised)

→ System discovers statistically salient features of i/p population (clustering)

Popularly used NN models

Some common features are true : but differ in fine details

→ Multi-layer perceptron

→ Hopfield's model ——

→

→

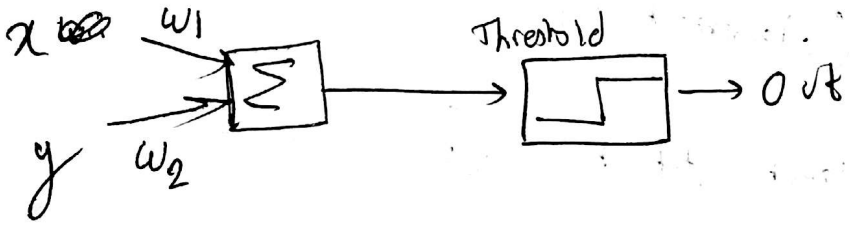
⋮

~~Applicability~~

→ Where human intell

Two input perceptron

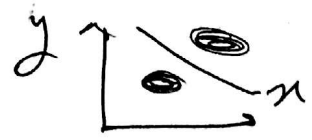
Perceptron : A single neuron connected by weights to a set of inputs



→ Let x & y be two inputs & w_1, w_2 be the weights

→ If $w_1 x + w_2 y > \theta$, then the output is 1 else 0, where θ = threshold

→ $w_1 x + w_2 y = \theta$ → separating line



Scanned by CamScanner

Learning rule

Learning : Present a set of input patterns, adjust the weights until the desired o/p occurs for each of them

$$w_i(t+1) = w_i(t) + \Delta_i ;$$

$$\Delta_i = \eta \, \delta \, x_i ;$$

$$\delta = T - A \quad (\text{i.e. target} - \text{actual})$$

→ If the sets of patterns are linearly separable, the single layer perceptron algo is guaranteed to find a separating hyperplane in a finite no. of steps.

Chng of weights