Distinguishing characteristics of AI — 16/9/19

- 2nd half

→ predicate calculus
→ inferences — resolution, refutation } on clauses.

question: i) given a set of sentences, can we prove a given sentence?

ii) uncertainty maintenance

→ game playing
→ neural networks (~~NN~~ study next week), single layer perceptrons.

AI Production system:

shows a separation between
data, operation, control

select a path among a set of alternatives
rule selection

Control ~~selection~~ strategy

- irrevocable
- tentative
  - backtracking
  - graph search control.

eg: 8 puzzle problems (4 rules)

Any search problem can be characterised by 4 ~~tg~~ tuples:

{ Start node, ~~End node~~, ~~state~~ Whole node, Transition function, Intermediate node }

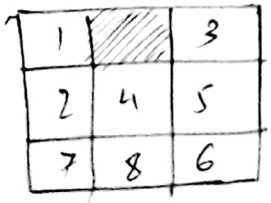
procedure production :

1. data = initial database / variable
2. until data satisfies termination condition 2 :

begin :
- select some rule R in the set of rules that can be applied to data
- data is the result of applying R to data.

---

8 puzzle ~~to~~ problem :



1. replace with left
2. replace with right
3. replace with bottom

} Control strategy determines which one is to be chosen

* replace with top inapplicable

FOPL :

Syntax : alphabet of symbols and how wff is formed.
well formed formula

Components : predicate symbol
var "
function "
constant "

Nielson Rhode - Principles of A-I

- Write (Nielson, Principles of AI) // Predicate Calculus

  Write (X, Principles of AI) → Out : X = N. R.

  Write (X, Y)

- John's brother & John's sister are sibling's to each other

~~Sibling (~~

~~Brot~~

~~Sibling (~~.

- ~~X~~ = Brother (John)
- Y = Sister (John)
- Sibling (X, Y)

Sol

Brother (John) // function
Sister (John) // function

Sibling (Brother (John), Sister (John))

- The house is yellow.

constant.

→ yellow (House 1). // predicate
or
color (h1, yellow)

Connectives : 1) conjunction
2) negation
3) disjunction
4) implication

- John lives in a yellow house

~~lives (John, h) ∧ house (~~

~~cat lives (John, h1)~~

~~live~~ lives (John, h1) ∧ color (h1, yellow)

~~class color (~~

John lives in yellow house

$$\boxed{\text{lives}(\text{John}, h1) \wedge \text{color}(h1, \text{yellow})}$$

John plays chess as player badminton

$$\boxed{\text{plays}(\text{John}, \text{chess}) \vee \text{plays}(\text{John}, \text{Badminton})}$$

If car belongs to John, then its grey.

$$\boxed{\text{Belongs}(\text{John}, c1) \Rightarrow \text{color}(c1, \text{grey})}$$

- All elephants are grey

$$(\forall x)\ \text{Elephant}(x) \Rightarrow \text{color}(x, \text{grey})$$

- There is a person who wrote principles of AI

$$\boxed{(\exists x)\ \text{person}(x) \wedge \text{write}(x, \text{pAI})}$$

- Whenever element exists Variables are bounded by quantifiers (bounded variable)

Propositional logic → FOPL, FO Predicate logic

↓

So Predicate logic.

(considers quantifiers for functions)

$$\sim(\exists x) P(x) \equiv \forall x (\sim P(x))$$

For every set X — there is set Y such that cardinality of Y is greater than X

~~$(\forall x)\ set(x) :-$~~

~~$(\forall x)\ set(x) \rightarrow ($~~

~~$(\forall x)\ set(x) \Rightarrow (\exists y)(set(y) \wedge card(y) \& card(x))$~~

$(\forall x)\ set(x) \Rightarrow (\exists y)(set(y) \wedge greater(card(y), card(x)))$

$(\forall x)\ set(x) \Rightarrow (\exists y)(\exists u)(\exists v)(set(y) \wedge card(x,u) \wedge card(y,v) \wedge greater(v,u))$

Rules of inference

1) $w_1 \Rightarrow w_2$

w_1 true ..... w_2 true Modus ponens.

2) Universal specialisation

$(\forall x)\ (w_1(x) \Rightarrow w_2(x))$

$w_1(A)$ = true, implies $w_2(A)$ = true

3) Substitution $\longrightarrow \{A/x\}$

3) $\{\ \} \xrightarrow{R_1} \{\ \} \xrightarrow{R_3}$

path of a solution

$\{wff\}$

proof

Unification : finding substitutions of terms for variables

t/v
↳ term can be substituted by v
→ c
→ f , provided does not have v

P[x, f(y), B]

P[z, f(w), B] — S{z/x, w/y}

P[x, f(A), B] — S{A/y}

P[g(z), f(w), B] — S{g(z)/x, w/y}

P[c, f(A), B] — S{c/x, A/y}

alphabetic variable — variable/variable
ground instance — const/variable

P( ). s_1 = { }

E. s (Notation)
↳ substitution
expression

E s_1 s_2 (Composition)

{g(x,y)/z} {A/x, B/y, C/w, D/z}

S1 S2

Not taken since already subs.

~~t/v~~ ~~g(A,B)/z~~ D

~~g(A,B)/z~~ w

~~g(A,B)~~

{g(A,B)/z, A/x, B/y, C/w}

→ Substitution applied by applying S2 to the terms of S1, and add any pair of S2 having variables not occuring in the variables of S1

$(S_1, S_2)\ S3 \rightleftarrows S1\ (S_2, S_3)$

same

- Composition of substitution is associative

$S_1 S_2 \neq S_2 S_1$

- They are not commutative

$E = \{ P[u, f(y), B],\ P[u, f(B), B] \}$

$s = \{ \quad , \quad \}$

$s = \{A/u, B/y\}$ → unifier

If I find an s such that set of expression becomes singleton set"

- If all the substitutions are not necessary for singleton set, then s is not most general unifier

$E = \{ P(u), P(A) \}$

$s = \{A/u\}$

$\{ P[f(u), y, g(y)],\ P[f(u), z, g(x)] \}$

$s = \{y/z\}$ (MGU)

Singleton set : $P[f(u), y, g(y)]$

$s = \{u/z,$

$s = \{y/z,$

$y/u\}$

MGU

Disagreement Set :

Unification algorithm iterature

Step 1: Set $k = 0$ ~~and MGU at k = null~~

$MGU(k) = null$

Step 2: If set MGU_k is a singleton.

then stop ; MGU_k is ~~not~~ MGU of F

~~If~~

otherwise

find disagreement set d_k of E MGU_k

Step3:

If there is variable v and term t in D_k such that

v does not occur in t, put MGU_{k+1} ~~= MGU_k~~

$= MGU_k \{t/v\}$

Set $k = k+1$ and return to step 2.

otherwise

stop

E is not unifiable

---

* Nilsan - 2nd version

$E = \{P(x,z,y), P(\omega,v,\omega), P(A,v,v)\}$

~~$D_0 = \{x, \omega, A, z, v, y, \omega, v\}$~~

$D_0 = \{x, \omega, A \quad z, v \quad y, \omega, v\}$

$s = \{A/x\}$

$D_1 = \{A, \omega \quad z, v \quad y, \omega, v\}$

~~$P(A,v,v)$~~

~~$A/x, A/\omega, v/z$~~

$s = \{A/x, A/\omega\}$

$D_2 = \{A, z, v, y, A, v\}$

$A/x, A/\omega, A/v, A/z, A/y\}$

$A, z, y, A, z \quad A/v, A/z$

$MGU = \{A/x, A/\omega, A/v, A/z, A/y\}$

Singleton $S = \{P(A,A,A)\}$

$\bar{A} \wedge \bar{B}$

Resolution Refutation Algorithm

23/10/19

WFF → base form

Resolution

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

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

1. Eliminate $\Rightarrow$ $\quad A \Rightarrow B : \sim(A \vee B)$
2. Reduce scope of negation.
3. Standardise var
4. Eliminate $\exists$
5. Convert to Prenex.
6. Put matrix in CNF
7. ~~Put matrix in CNF~~
7. Eliminate $\forall$
8. Separate into clauses.
9. Rename var.

~~$(\forall x)\{[P(x) \Rightarrow \forall x\, Q(x)$~~

~~$(\forall x)\{\sim(P(x) \wedge$~~

4)a) $(\forall y)[(\exists x)\, P(x,y)]$

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

skolem function

(Only when are ~~forall~~ $\forall$ precedes $\exists$)

4)b) If I just have $\exists x\, P(x)$ ~~then~~ (No ~~forall~~ $\forall$)

then ~~P(c)~~ $P(C)$

Skolem constant.

Skolemization (to remove existential quantifier)

5) Prenex form : All $\forall$ quantifiers are infront of expression

~~Prefix~~ is $\forall$ part is prefix

remaining part is matrix.

c) $u_1 \wedge (u_2 \vee u_3)$ ⑧

$- (u_1 \wedge u_2) \vee (u_1 \wedge u_3)$ ⑨

Example.

① $(\forall u)(\sim(P(u) \vee (\forall y [\sim(P(y) \wedge P(f(u,y))] \wedge \sim(\forall z)[\sim(Q(u,y) \wedge P(y))]))$ (③ done before)

② $(\forall u)(\sim(P(u) \vee (\forall y [\sim(P(y) \wedge P(f(u,y))] \wedge \exists z [Q(u,z) \vee \sim P(z)]))$.

③ $(\forall u)(\sim(P(u) \vee ((\forall y)[\sim(P(y) \wedge P(f(u,y))] \wedge (\exists z)[Q(u,z) \vee \sim P(z)]))$

④ $(\forall u)(\sim(P(u) \vee ((\forall y)[\sim(P(y) \wedge P(f(u,y))] \wedge [Q(u, h(u)) \vee \sim P(h(u))]))$

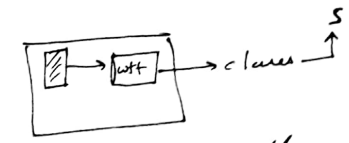
⑤ $(\forall u)(\forall y)\{\sim(P(u) \vee [\sim(P(y) \wedge P(f(u,y))] \wedge [Q(u, h(u)) \vee \sim P(h(u))]$

⑧ Replace $w_1 \wedge w_2$ into set of ~~clauses~~ wffs $\{w_1, w_2\}$

⑨ ~~Rename~~ Replace ~~the~~ variables.

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 $Q \vee \sim Q$	Tautology
$\sim P$ and P	NIL	empty clause ↓ Sign of contradiction
~~Parent~~ $\sim P \vee Q$ and $\sim Q \vee R$	$\sim P \vee R$	chain clause.

Resolution

⑩ 

Whether w logically ~~is~~ follows from S

* We would be using ~~Resolution~~ Refutation along with ~~resolution~~ to prove w.

• We append w to S

if $\{S \vee \sim w\}$ ultimately gives NIL, then w follows from S

• Which 2 parent clauses must be taken so that NIL is produced (search strategy)

Production System for RR

- Let S be a set of clauses (base set)
- Procedure Resolution :

step 1 : clauses ← S
step 2 : until NIL ∈ clauses
step 3 : begin
step 4 : • select 2 distinct resolvable clauses c_i and c_j from clauses.
step 5 : • compute a resolvent R_{ij} of c_i and c_j
step 6 : • set produced but adding R_{ij} to clauses.

Example:

1) Whoever can Read is literate
2) Dolphins are not literate
3) Some dolphins are intelligent

Prove that: Some who are intelligent cannot read.

a) $(\forall x)\big(Read(x) \Rightarrow Literate(x)\big)$

b) $\sim Literate(Dolphin(x))$. $Dolphin(x) \Rightarrow \sim Literate(x)$

c) $(\exists x)\,(Dolphin(x) \wedge Intelligent(x))$.

$(\exists x)\{$

a) $(\forall x)\ (R(x) \rightarrow L(x))$

b) $D(y) \rightarrow \sim L(y)$

c) $(\exists z)\ (D(z) \wedge I(z))$

d) $(\exists w)\ (I(w) \wedge \sim R(w))$

a) $\sim(R(x) \wedge L(x))$

b) $\forall y\,(\sim D(y) \vee \sim L(y))$

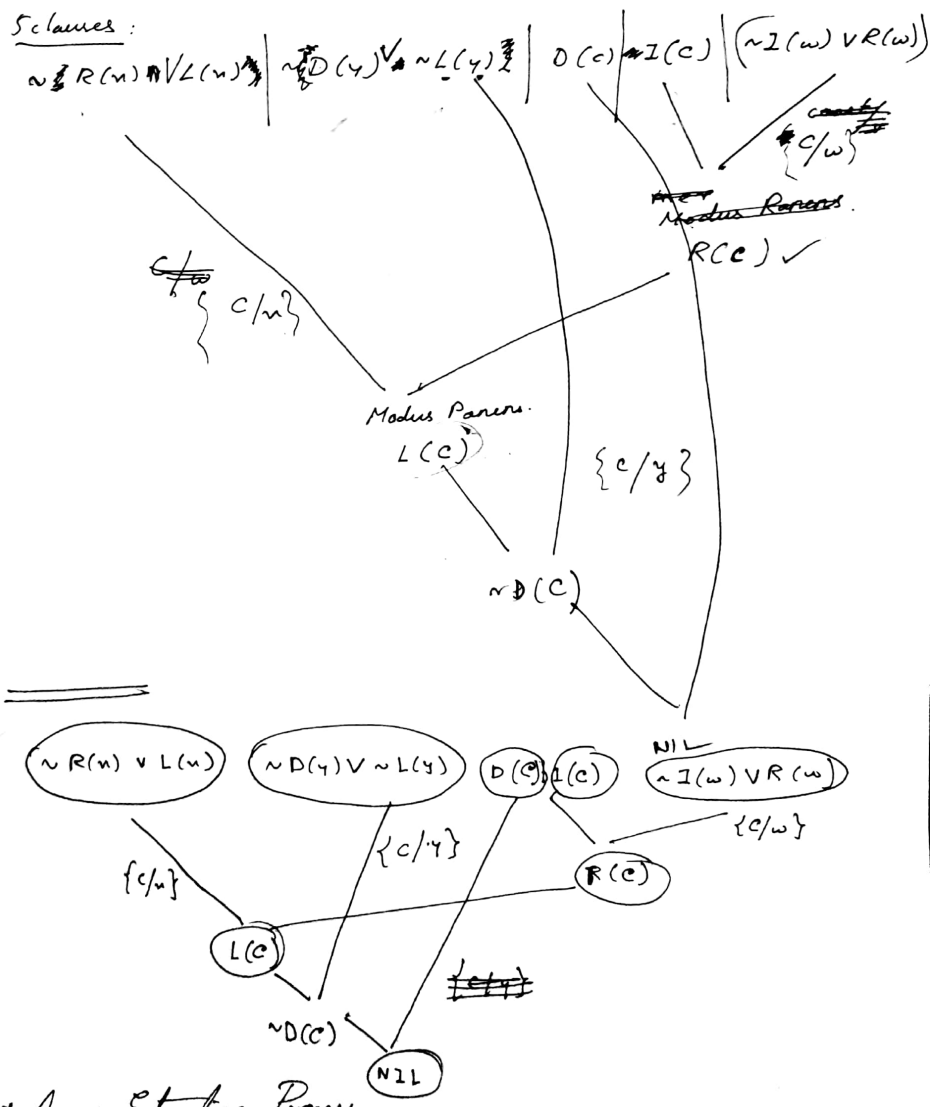
c) $D(c) \wedge I(c)$

d) ~~$\exists(c) \wedge \sim R(c)$~~

d) $\sim W = \sim (\exists w)\,(I(w) \wedge \sim R(w))$

$= (\forall w)\,(\sim I(w) \vee R(w))$

$= (\sim I(w) \vee R(w))$



* Answer Extraction Process

6/11/19

Control Strategy for Resolution Refutation:

1. **Breadth first**

   Compute all 1st level resolvent, then lvl 2, lvl 3, ...

   — Complete, inefficient // complete means whether control strategy could prove goal wff if exists.

2. **Linear Input form**

   each resolvent has at least 1 parent belonging to base sets

   (not complete)

3) **Set of Support** —

   At least 1 parent of each resolvent is selected from among the clauses resulting from negation of goal wff's or descendents (complete)

4) **Unit preference**

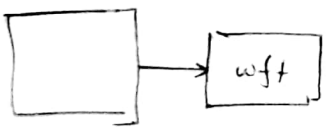
   try to select a single literal clause (unit) into be parent

   → No. of literals ↓ will increase or resolution

5) **Ancestry filtered form**

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

6) **Combination**



rules of inference → → — - - derived wff / goal wff.

(2) $\sim Q(u) \vee \sim P(u)$ $Q(y) \vee \sim P(y)$ $\sim Q(w) \vee P(w)$ $Q(v) \vee P(A)$

$\{u/y\}$

$\sim P(u)$

$\{v/u\}$

$\sim Q(u)$

nil allowed for case (2), but for case (5)

$P(A)$

nil

→ If Fido goes wherever John goes, and John goes to school, where is Fido.

① $(\forall x)[place(\underset{John}{\cancel{Fido}}, x) \Rightarrow place(\underset{Fido}{\cancel{John}}, x)]$

~~place (Fido, School)~~

② place (John, School)

③ $(\exists x)$ place (Fido, X)

a) ~place (John, X) V place (Fido, x)

b) place (John, School)

c). $(\exists x)$ place (Fido, X) goal wff. ~~Skolem~~

so ~w ~ $(\exists x)$ place (Fido, X) → $(\forall x)$ (~place (Fido, x))

~~To get Ans~~ ~~~place(J,x) V p(F~~

~p (J, x) V p (F, x) p (J, S) ~p (F, X)

{S/x}

p (F, S) {S/x}

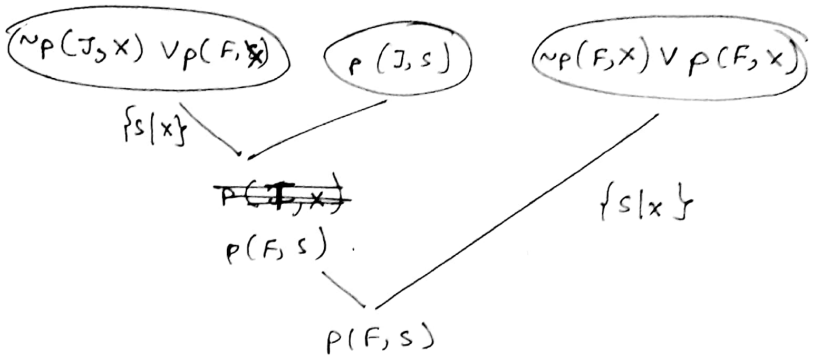
Nil

To get Answer: ① Append to each clause arising from negation of goal wff its own negation.

② Following structure of refutation tree perform same resolutions as before until same clause is obtained at the root.

③ Use the clause at the root as an answer statement

This tt tree is called "modified proof tree"



Modified Proof tree

6/22/19

AND-OR graph:

→ all nodes need to be processed
→ one of the nodes - - - - - - -

hypergraph

hyperarc.

K-Connector

Formulating game playing as search: —

2 person, perfect info

player moves alternatively, no chance factor

- Iterative methods
- Adversary

Evaluation fn of Tic tac toe.
f(n) = weighted features

Static evaluation function:

f(n) = large +ve
= large -ve.
= +∞
= -∞.

Game Tree

Root → decision on what is the best single move to make next.

my turn → MAX

else MIN

Arcs - possible legal moves.

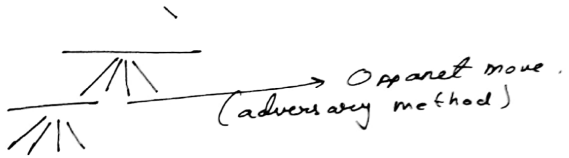
- In each level — MAX / MIN
- Nodes corresponding to MIN's next move have successors that are like AND nodes.
- Nodes - - - - - MAX's - - - - -

  - - - like OR nodes

Game — # nodes in complete game tree

Checkers 10^{40}

Chess 10^{120}

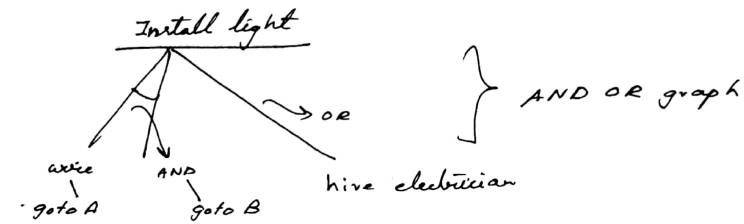


Tic tac toe — conventional search won't work. (OR-graph example)

→ If I have a wire and a screwdriver, I could install the light

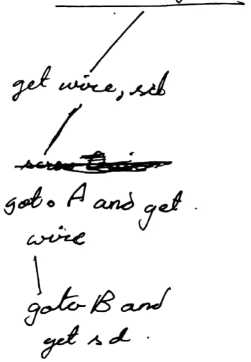
- hypergraph
- hyperarc

- Install

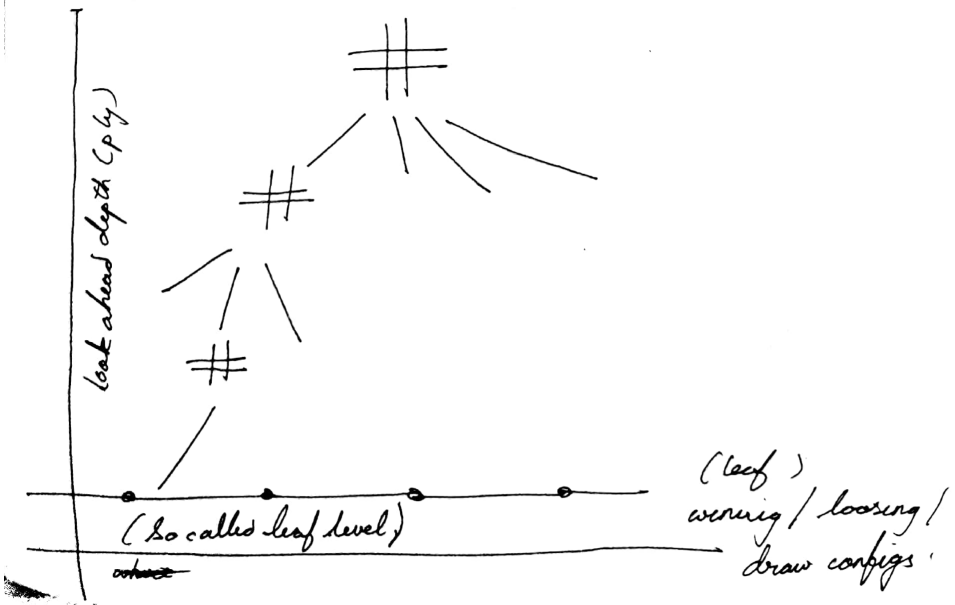


In 1 single graph, same as AND, same as OR

Removal of AND:



- This will be used for formulating 2 person games —



On each ~~node~~ leaf node, a static evaluation function is applied.

If $f(n)$ = large +ve, it is a ~~winning config~~ favourable according to my configs

$f(n)$ =

$= +\infty$ // really won

$= -\infty$ // really lost

Eval function for me = no. of 3 lines open for me

— - - - - - - : opponent

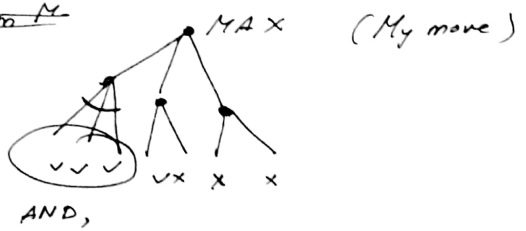
For chess:

weighted features

$w_1 f_1 + w_2 f_2 + \cdots$

Game tree

~~From M~~

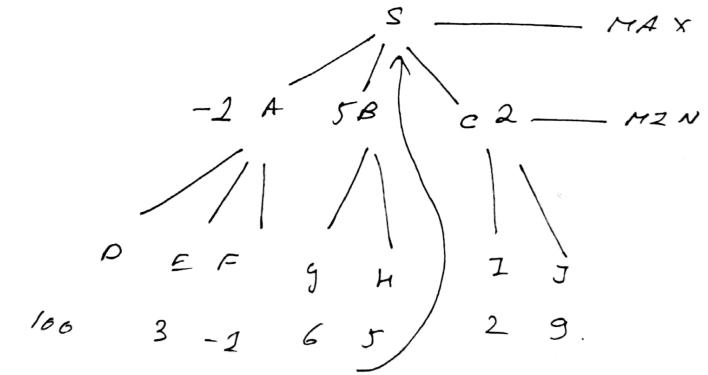


AND,

I have to consider the move such that ~~ANDing~~ all moves my opponent makes ~~does~~ lead to my winning

Searching game tree using MINI-MAX

1. Create start node as MAX node (my turn to move) with current board config
2. Expand nodes upto some depth (ply) ~~of~~ or leaf nodes
3. Apply evaluation func. at leaf
4. Backup values at each of the non-leaf nodes until values are computed at root.
5. At MIN node, backed up value is minimum of values associated with successor
6. At MAX node, max of values associated with child nodes
7. Pick operator associated with child node node ~~with~~ that whose backed up value determined value at root.

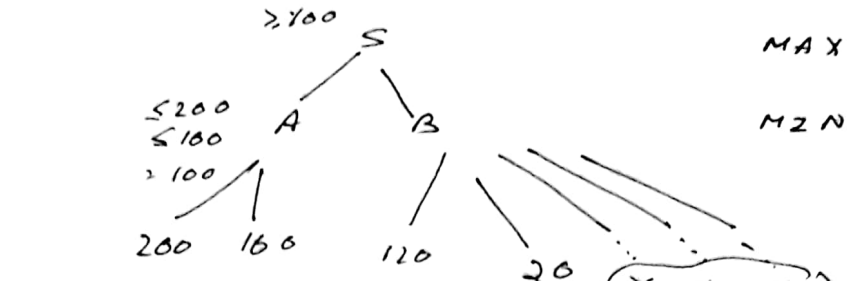


Path

S → B → H.

- Although 100 gives max winning score we do not choose A, ~~so~~ because opponent ~~at~~ can go to F.

α-β pruning :

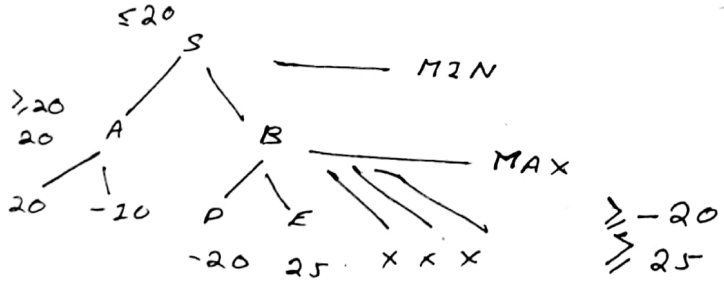


α- cutoff (MIN)

α- Best value

can be pruned because after 20 these branches do not contribute to S score.

β-cutoff (MAX)



Chess — 35 (Branching factor is reduced to 6)

↓

6

Worst case → b^d.

best → $(2b)^{d/2}$

* Horizon effect : (Lookahead problem)

Uncertainty Handling (Gap)

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

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

Out-nodes → OUT, ... None

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

Normal

deduction:

ANN (Neurocomputing Intro)

(Connectionist Model

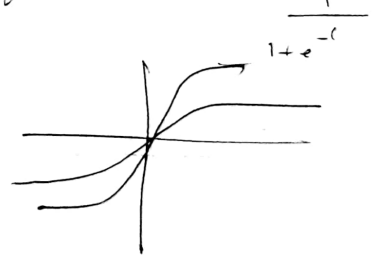
Parallel distributed Processing (PDP) // Rumel Hart
(protein secondary structure)

→ Hopfield Nets // clustering (≈ unsupervised)

• Multilayer Perceptron → classification

sigmoid: $\frac{1}{1+e^{-\frac{(u-\theta)}{\theta_0}}}$

slope



• sequential vs batch training

• If non-linearly separable, train