Rules of Inference

1. $P \wedge Q \Rightarrow P$, $P \wedge Q \Rightarrow Q$ } Simplification.
2. $P \Rightarrow P \vee Q$, $Q \Rightarrow P \vee Q$ } Addition
3. $P, Q \Rightarrow P \wedge Q$ Conjunction
4. $\neg P, P \vee Q \Rightarrow Q$ Disjunctive Syllogism
   $\hookrightarrow P, \neg P \vee Q \Rightarrow \neg Q$.
5. $P, P \to Q \Rightarrow Q$ Modus Ponens.
6. $\neg Q, P \to Q \Rightarrow \neg P$. Modus Tollens.
   $\hookrightarrow Q, P \to \neg Q \Rightarrow \neg P$
7. $P \to Q, Q \to R \Rightarrow P \to R$ Hypothetical Syllogism
8. $P \vee Q, \neg P \vee R \Rightarrow Q \vee R$ Resolution.

Logical Equivalences

1. $P \wedge T \Rightarrow P$. $P \vee T \Rightarrow T$
2. $P \vee F \Rightarrow P$ $P \wedge F \Rightarrow F$.
3. $P \vee P \Rightarrow P$. $P \wedge P \Rightarrow P$
4. $p \vee q \Rightarrow q \vee p$ $p \wedge q \Rightarrow q \wedge p$.
5. $(p \vee q) \vee r \Rightarrow p \vee (q \vee r)$ $(p \wedge q) \wedge r \Rightarrow p \wedge (q \wedge r)$
6. $p \vee (q \wedge r) \Rightarrow (p \vee q) \wedge (p \vee r)$.
7. $p \wedge (q \vee r) \Rightarrow (p \wedge q) \vee (p \wedge r)$.
8. $\neg(p \wedge q) \Rightarrow \neg p \vee \neg q$ $\neg(p \vee q) \Rightarrow \neg p \wedge \neg q$
9. $p \vee (p \wedge q) \Rightarrow p$ $p \wedge (p \vee q) \Rightarrow p$
10. $P \vee \neg P \Rightarrow T$ $P \wedge \neg P \Rightarrow F$
11. $p \to q \Rightarrow \neg p \vee q$ $p \vee q \Rightarrow \neg p \to q$.
12. $p \to q \Rightarrow \neg q \to \neg p$
13. $p \wedge q \Rightarrow \neg(p \to \neg q)$
14. $\neg(p \to q) \Rightarrow p \wedge \neg q$.
15. $(p \to q) \wedge (p \to r) \Rightarrow p \to (q \wedge r)$
16. $(p \to r) \wedge (q \to r) \Rightarrow (p \vee q) \to r$.
17. $(p \to q) \vee (p \to r) \Rightarrow p \to (q \vee r)$
18. $(p \to r) \vee (q \to r) \Rightarrow (p \wedge q) \to r$.
19. $p \leftrightarrow q \Rightarrow (p \to q) \wedge (q \to p)$
20. $p \leftrightarrow q \Rightarrow \neg p \leftrightarrow \neg q$

Convert statement into CNF:

1. Eliminate implications & biconditionals
   $P \leftrightarrow Q \Rightarrow (P \to Q) \wedge (Q \to P)$ & $P \to Q \Rightarrow \neg P \vee Q$.
2. Apply De Morgan's Law to reduce not symbols
3. use distributive & other laws to obtain CNF: $(P \vee R) \wedge (\neg Q \vee R) \wedge (R \wedge Q)$

Rules of Inference for Quantifiers:

Universal Generalization: $P(m) \Rightarrow \forall x (P(x))$.

Existential Generalization: $P(m) \Rightarrow \exists x (P(x))$

Universal Instantiation: $\forall x (P(x)) \Rightarrow P(m)$.

Existential Instantiation: $\exists x (P(x)) \Rightarrow P(m)$.

Also,

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

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

Conversion to CNF:

1. Eliminate biconditionals & implications
2. Move $\neg$ (negation) inwards.
3. Standardize variables by renaming them
4. Skolemize: Replace ~~Existential~~ $\exists$ variable with a Skolem function
5. Drop all $\forall$ variables.
6. Distribute $\wedge$ over $\vee$:

- If $p \to q$ & $\neg q \to \neg p$ contrapositive
- If $p \to q$ & $q \to p$. converse
- If $p \to q$ & $\neg p \to \neg q$ inverse

Eg:- Everyone who loves all animals is loved by someone.

$\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x,y)] \Rightarrow [\exists y \text{ Loves}(y,x)]$

1. $\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Loves}(x,y)] \vee [\exists y \text{ Loves}(y,x)]$
2. $\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Loves}(x,y)] \vee [\exists y \text{ Loves}(y,x)]$
3. $\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Loves}(x,y)] \vee [\exists z \text{ Loves}(z,x)]$
4. $\forall x [(\text{Animal } F(x)] \wedge \neg(\text{Loves}(x, F(x))] \vee \text{Loves}((G(x), x))$.
5. $[\text{Animal } F(x) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$
6. $[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)$

① $\forall x\ G(x) \rightarrow H(x)$ ② $\forall x\ H(x) \rightarrow S(x)$ ③ $G(John)$ ④ $\neg S(John)$

Eliminate $\rightarrow$:

① $\forall x \neg G(x) \vee H(x)$ ② $\forall x\ \neg H(x) \vee S(x)$ ③ $G(John)$ ④ $\neg S(John)$.

Move $\neg$ inwards. (not needed) Standardize variables:

① $\forall x\ \neg G(x) \vee H(x)$ ② $\forall y\ \neg H(y) \vee S(y)$ ③ $G(John)$ ④ $\neg S(John)$.

Skolemize (not needed) Drop all $\forall$:

① $\neg G(x) \vee H(x)$ ② $\neg H(y) \vee S(y)$ ③ $G(John)$ ④ $\neg S(John)$

Distribute $\wedge$ over $\vee$ (not needed). Resolution:

from ②, ④ $\Rightarrow$ $\neg S(John)$, $\neg H(y) \vee S(y) \Rightarrow \neg H(John)$. — ⑤ $\{y/John\}$

from ①, ⑤ $\Rightarrow$ $\neg H(John)$, $\neg G(x) \vee H(x) \Rightarrow \neg G(John)$ — ⑥ $\{x/John\}$

from ③, ⑥ $\Rightarrow$ $G(John) \vee \neg G(John) \Rightarrow F$ $\{z/John\}$.

(Predicate Logic → Truth Table
Propositional Logic → functions & objects).-

Unification

Unify (Knows (John, x), knows (John, Jane) ⇒ {x | Jane}.

Unify (knows (John, x), knows (y, Bill) ⇒ {y / John, x / Bill}.

Unify (knows (John, x), knows (y, mother(y)) ⇒ {y / John, x / mother(John)}

Unify (knows (John, x), knows (x, Elizabeth) ⇒ Fail. {x / Elizabeth,

↳ After standardization: knows (John, x), knows (z, Elizabeth) = y / John}.

—x—

- Some ppl who likes chocolates are tall: $\exists x\ (likes(x, chocolates) \wedge tall(x))$
- Everyone is loyal to someone: $\forall x \exists y\ loyalto(x, y)$.
- People only try to kill rulers they are not loyal to: $\forall x \forall y\ (ppl(x) \wedge ruler(y)) \rightarrow kill(x, y) \rightarrow \neg loyal(x, y)$

Given: ① man(marcus) ② roman(marcus) ③ ¬man(x), person(x)

④ ¬roman(x), loyal(x, caesar), hate(x, Caesar) ⑤ ruler(Caesar).

⑥ loyal(x, f(x)) ⑦ ¬person(x), ¬ruler(y), ¬kill(x, y), ¬loyal(x, y).

⑧ kill(marcus, caesar). RESOLUTION PROOF

Forward Chaining

Using Rules, prove fact

Backward chaining

Using facts, prove rules.

¬hate(marcus, caesar)   hate(x, Caesar).

{marcus | x}

man(x) (marcus)   ¬roman(x) ∨ loyal(Marcus, Caesar);

{Marcus | x}

¬man(marcus) ∨ loyal(marcus, Caesar).

2

7   loyal(marcus, Caesar).

{marcus / x, caesar / y}

1   ¬man(marcus) ∨ ¬ruler(caesar) ∨ ¬kill(marcus, Caes

ruler(caesar) ∨ ¬kill(marcus, Caesar). 4

¬kill(marcus, Caes) 8

F

HILL CLIMBING

- only care about cost of solution not path to the solution.
- heuristic search method to check how close a given state is to its goal state
- uses Greedy approach (~~loop~~ until best move found, then quit).
- no backtracking as it does not remember prev states.
- solution may not be optimal or complete.
- variant of generate & test algo - eg - Travelling Salesman Problem

Limitations : Local Maxima, Plateau, Ridge{ steep slope where search direction is not top-facing

~~final state~~

1	2	3
4	5	6
7	8	

~~initial state~~ (no. of mismatched tiles)

1	2	3
	4	6
7	5	8

depth $g = 0$
$h = 3$ ← manhattan distance
$f = 3$ — min.

1	2	3
4		6
7	5	8

$g = 1$
$h = 2$
$f = 3$ - min.

1	2	3
7	4	6
	5	8

$g = 1$
$h = 4$
$f = 5$

	2	3
1	4	6
7	5	8

$g = 1$
$h = 4$
$f = 5$

1		3
4	2	6
7	5	8

$g = 2$
$h = 3$
$f = 5$

1	2	3
4	6	
7	5	8

$g = 2$
$h = 3$
$f = 5$

1	2	3
4	5	6
7		8

$g = 2$
$h = 1$
$f = 3$ min

1	2	3
4	5	6
	7	8

$g = 3$
$h = 1$
$f = 4$

1	2	3
4	5	6
7	8	

$g = 3$
$h = 0$
$f = 3$
min = goal
STOP.

SIMULATED ANNEALING (PROBABILISTIC/STOCHASTIC HILL CLIMBING)

- finds approx global maxima & avoids being caught up in local ~~minima~~ maxima.
- yields efficiency & completeness

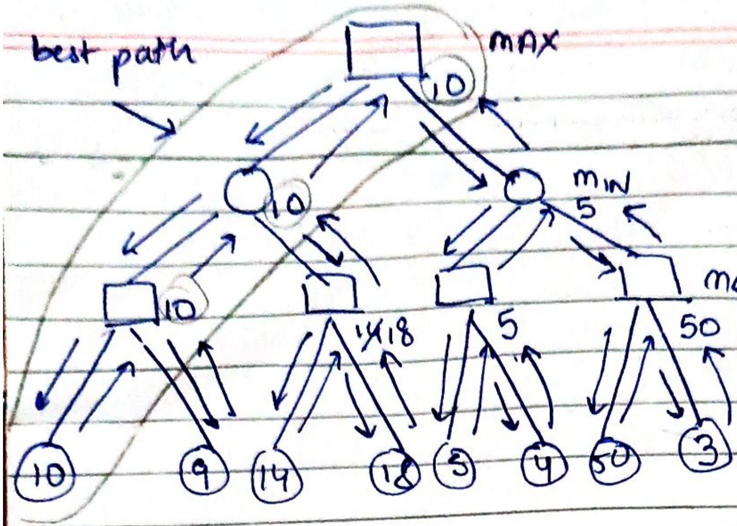
Algo: Select one neighbour randomly. Find $e^{\Delta E/T}$ of current & neighbour node. If e ~~of~~ of next node greater than current node ($\Delta E > 0$) select neighbour node as next node, else find selection probability based on exponential formula and select resulting node.

minimax (DFS + Depth Limited Search)

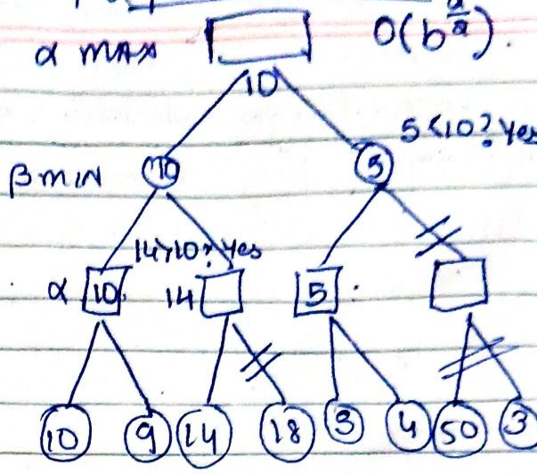
- recursive backtracking algo used in decision making
- provides optimal sol. - used for 2player game playing like chess & tic-tac-toe
- chance of one person winning the game is possible at the expense of other person loosing the game.
- Start at curr position & use a move generator to predict possible successor positions. Apply evaluation func() for those positions & chose

the best one. After doing so, we can back that value up to start position and evaluate it.

- Complete, optimal, Time Complexity: $O(b^d)$, Space Complexity: $O(bd)$.
- MAX = player, MIN = opponent.



αβ PRUNING



GENETIC ALGO

1. Initialization: initializing a population (set of chromosomes / solutions) typically done to provide even coverage of search space. A chromosome (string of genes) is characterized by genes.
2. Evaluation: population is evaluated by assigning a fitness value to each individual. In this stage, note current & avg fitness sol.
3. Selection: if termination condition is not met, population goes thru selection stage where individuals are selected based on fitness score. (methods of selection: Roulette wheel, Rank selection, Steady state, Tournament)
4. Reproduction: selecting parents from desired population to produce new offspring using crossover (ensures that ~~each~~ search progresses in the right direction by making new chromosomes that posses similar characteristics to both parents) & mutation (maintain diversity in population). Types of Crossovers:-

- 1 point: parents { 111111|11, 000000|00 } children { 11111100, 00000011 }
- 2 point: parents { 11|1111|11, 00|0000|00 } children { 00111100, 11000011 }
- uniform: parents { 1111111111, 0000000000 } children { 0010111001, 1101000110 }
  mask: 1110 1000110
- mutation: before 0010111001 → after → 0000111010

x_i	Xd	$F(x)=x^2$	$P=\frac{F(x_i)}{\Sigma x}$	$E=\frac{F(x_i)}{\bar{x}}$	Actual Count	Mating Pool	Crossover	Mutation	Offsprings
01101	13	169	0.14	0.58	1	0110|1	01100	x	01100
11000	24	576	0.49	1.97	2	1100|0	11001	111001	111001
01000	8	64	0.06	0.22	0	11|000	11011	x	11011
10011	19	361	0.31	1.23	1	10|011	10000	x	1000

$\Sigma = 1170$

$\bar{x} = 292.5$