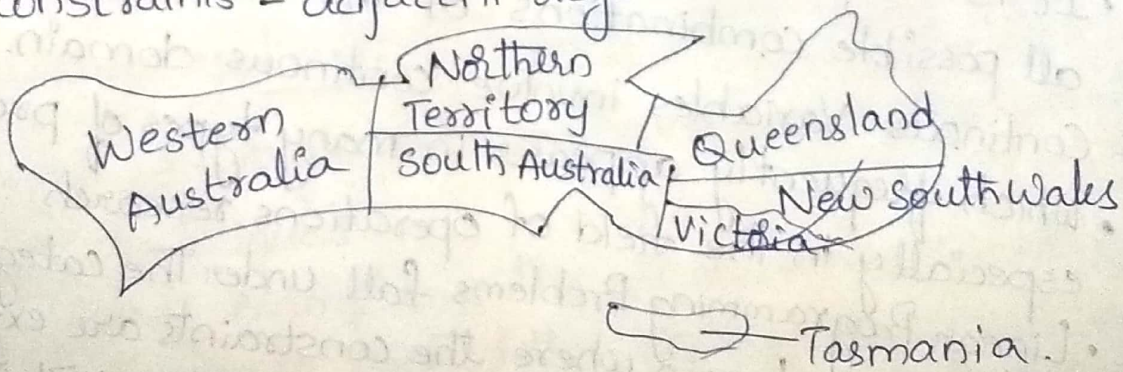
Constraint Satisfication Problem

- A problem described by imposing constraints on the variables related to a problem is called Constraint Satisfication Problem (CSP)
- A CSP consists of 3 components X, D & C.
  - X is a set of variables $\{x_1, x_2 \cdots x_n\}$ variable.
  - D is a set of domains $\{D_i, D_{i+1} \cdots D_n\}$, one for each
  - C is a set of constraints that specify allowable combination of values.
  - Each domain D_i consists of a set of allowable values $\{v_1, v_2 \cdots v_n\}$ for variable x.
  - Each constraint C_i consists of a tuple of variables participating in the constraint.
  - A constraint is a relation defining the values that variable can take on.

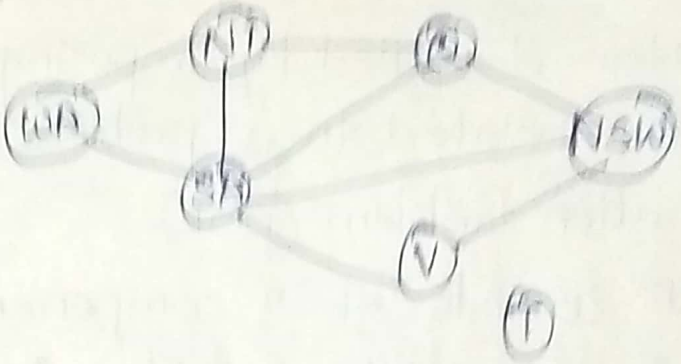
Example – Map-Coloring Problem

- Variables $X = \{WA, NT, Q, NSW, V, SA, T\}$
- Domain $D_i = \{red, blue, green\}$
- Constraints – adjacent regions must have different color



- Solutions are complete & consistent assignments (WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green)

Constraint graph = nodes are variables, arcs/edges are constraints.



Types of Variables

- Discrete variables with finite domains (Color, graphing, & Queens)
  - If the domain size of any variable is d, then the possible no of complete assignments will be $O(d^n)$, where n is no of variables.
  - Finite domains include Boolean values.
- Discrete variables with infinite Domains.
  - Infinite domains are represented in terms of integers & strings
  - The no of values that can be assigned to a variable could be infinite.
  - It is not possible to define constraints by considering all possible combinations of the values.
- Continous Variables involve continous domain.
  - Which frequently appears in many types of problems, especially in the field of operations research.
  - Linear Programming Problems fall under the category of continuous domain where the constraints are expressed using linear in-equalities (eg $x+y>5$, $2x-y>1$, $8 \leq 3-x<12$)

Types of Constraints

- Linear Constraints – Linear combination of variables expressed in Mathematical expressions (Linear equations, linear inequalities, logical Expressions)
- Non-Linear Constraints – (Polynomial, Exponential, Trignometric & Logarithmic)
- Absolute Constraints – Constraints are imposed on the variables using either absolute values & inequalities
- Preference Constraints – Constrainst imposed based on the user preferences.

Classification of Constraints

- Unary Constraints – involves single variables
- Binary Constraints – involves two variables
- High-Order constraints – involves three & more variables

Example application implemented using CSP

- solve cryptographic arithmetic puzzles using CSP techniques.
- A classic example is the SEND+MORE=MONEY puzzle, where each letter represents a unique digit(0-9)

- Each letter stands for a distinct digit.
- The aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct with the added restriction that no leading zeros are allowed.

Variables – F, T, O, U, W, R

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$

Constraints

- No repeated digits
- Only one carry forward allowed.
- Problem can be solved from both sides i.e lefthand side (LHS) & righthand side (RHS)

Step 1

- Starting from LHS are T & T. Assign a digit which should give a satisfactory result.

Let's assign T → 9.

$$\begin{array}{r} T \\ +\ T \\ \hline F\ O \end{array} \qquad \begin{array}{r} T\,(9) \\ +\ T\,(9) \\ \hline (1)\,F\ O\,(8) \end{array}$$

$$\begin{array}{r} (9)\ T\ \overset{(3)}{W}\ O\ (8) \\ +\ \overset{(9)}{T}\ \overset{(3)}{W}\ O\ (8) \\ \hline F\ O\ U\ R\ (6) \\ (1)\ (8)\ (7) \end{array}$$

T – 9
W – 3
O – 8
F – 1
U – 7
R – 6

Step2

- Now, Move ahead to the next terms W & W to get U as its o/p.

$$\begin{array}{r} W-3 \\ +\ W-3 \\ \hline U-6 \end{array}$$

– not possible because we cannot assign the same digit to two letters i.e U & R. as O – 8

- Add carry 1 to the value U to change the value of alphabet

$$\begin{array}{r} W \\ +\ W \\ \hline U \end{array} \qquad \begin{array}{r} \overset{1}{3} \\ +\ 3 \\ \hline 7 \end{array}$$

Step3

Next, consider the remaining letters

$$\begin{array}{r} O-8 \\ +\ O-8 \\ \hline R\ (16) \end{array}$$

carry

Example 1

- Given a cryptarithmetic problem i.e SEND + MORE = MONEY.

Step 1

- Starting from the left hand side (LHS), the terms S & M. Assign a digit which could give a satisfactory result.

```
  S          S - 9
+ M        + M - 1
----       -------
 M O         M O
carry (1)   (1) (0).
```

- Hence, we get a satisfactory result by adding the terms & got an assignment for O as O → 0 (zero) as well.

Step 2

- Now, move ahead to the next terms E & O to get N as its o/p

```
                          (1) - carry.
  E   ✗→   5         E   ——→   5
+ O      + O       + O          O
---      ---       ---        ---
  N        5         N          6
```

- Adding E & O which means 5+0=5, which is not possible because we cannot assign the same digit to two letters
- Add Carry 1 to the value E to change the value of alphabet

Step 3

- Further, adding the next two terms N & R we get,

Scanned with CamScanner

1 - carry.

$$\begin{array}{r} N \\ +R \\ \hline E \end{array} \rightarrow \begin{array}{r} 6 \\ +8 \\ \hline 14 \end{array} \times \qquad \begin{array}{r} N \\ +R \\ \hline E \end{array} \quad \begin{array}{r} 6 \\ +8 \\ \hline 15 \end{array}$$

- But we have already assigned E→5. Not possible with 5 to E. Again after solving the whole problem, we will get a carryover on this term, so our result will be satisfied

Step-4

Again, on adding the last two terms i.e the rightmost terms D & E, we get Y as its result

$$\begin{array}{r} D \\ +E \\ \hline Y \end{array} \rightarrow \begin{array}{r} 7 \\ +5 \\ \hline 12 \end{array}$$

- Keeping all the constraints in mind, the final resultant is as follows.

$$\begin{array}{r} SEND \\ +MORE \\ \hline MONEY \end{array}$$

Alphabet	digit
S	9
E	5
N	6
D	7
M	1
O	0
R	8
Y	2

Components of CSP Based Search

(1) Initial State - the empty assignment in which all variables are unassigned.

(2) Successor function - a value can be assigned to any unassigned variable, if it does not conflict with a previously assigned value.

(3) Goal Test - The current assignment is complete

(4) Path cost - A constant cost of 1 for every step

- Some Observations
  - The search is complete
  - The sol appears at depth n where n is the no of variables existing in the problem.
  - $b=(n-1)d$ at depth l.

CSP Search Algorithms

(a) Forward Search. - steps

(1) Initialize - start with the initial assignment of variables & their domains. Create an empty set to store the list of constrained variables.

(2) Select Variable - Choose a variable to assign a value that can be based on heuristic like Minimum Remaining Value (MRV) or Degree Heuristic

(3) Select Value - Choose a value from the domain of the selected variable based on heuristic like least Constrained Value (LSV).

(4) Assign Value - Assign the selected value to the selected variable

(5) Update domain - for each unassigned variable adjacent to the assigned variable, remove the selected value from their domain.

(6) Check Domain Emptiness – If any domain becomes empty, backtrack to the previous variable & try a different value.

(7) Check for Solution – If all variables are assigned values & all constraints are satisfied, you have found a solution.

(8) Recursive Step – If not all variables are assigned, recursively repeat steps 2-7 for the next variable.

(9) Backtrack – If you reach a point where no value can be assigned to the current variable, backtrack to the previous variable & try a different value.

(10) Return Solution – If a solution is found then return it If you reach a point where all possible assignments have been tried & none succeed, report failure.

(11) Undo assignments – Before backtracking, undo the assignment of the current variable & restore the domains of variables that were updated.

(12) Continue Search – Continue the search process until a solution is found or all possible assignments have been tried.

Minimum Remaining Value (MRV)

The main idea behind this is its choose the variable with the fewest "legal" values.

Eg after the assignments for WA = red & NT = green, there is only one possible value for SA, so it makes sense to assign blue to SA next rather than Q.

Selecting variables with highest degree

The main idea behind this is it selects the variable involved in the largest no of constraints on other unassigned variables.

Eg SA is the variable with highest degree 5, the other variables have degree 2 & 3 except for T which has 0 so it selects SA variable first.

- The MRV heuristic is usually a more powerful guide, but the degree heuristic can be useful as a tie-breaker.
- Once a variable has been selected, the algorithm must decide on the order in which to examine its values.

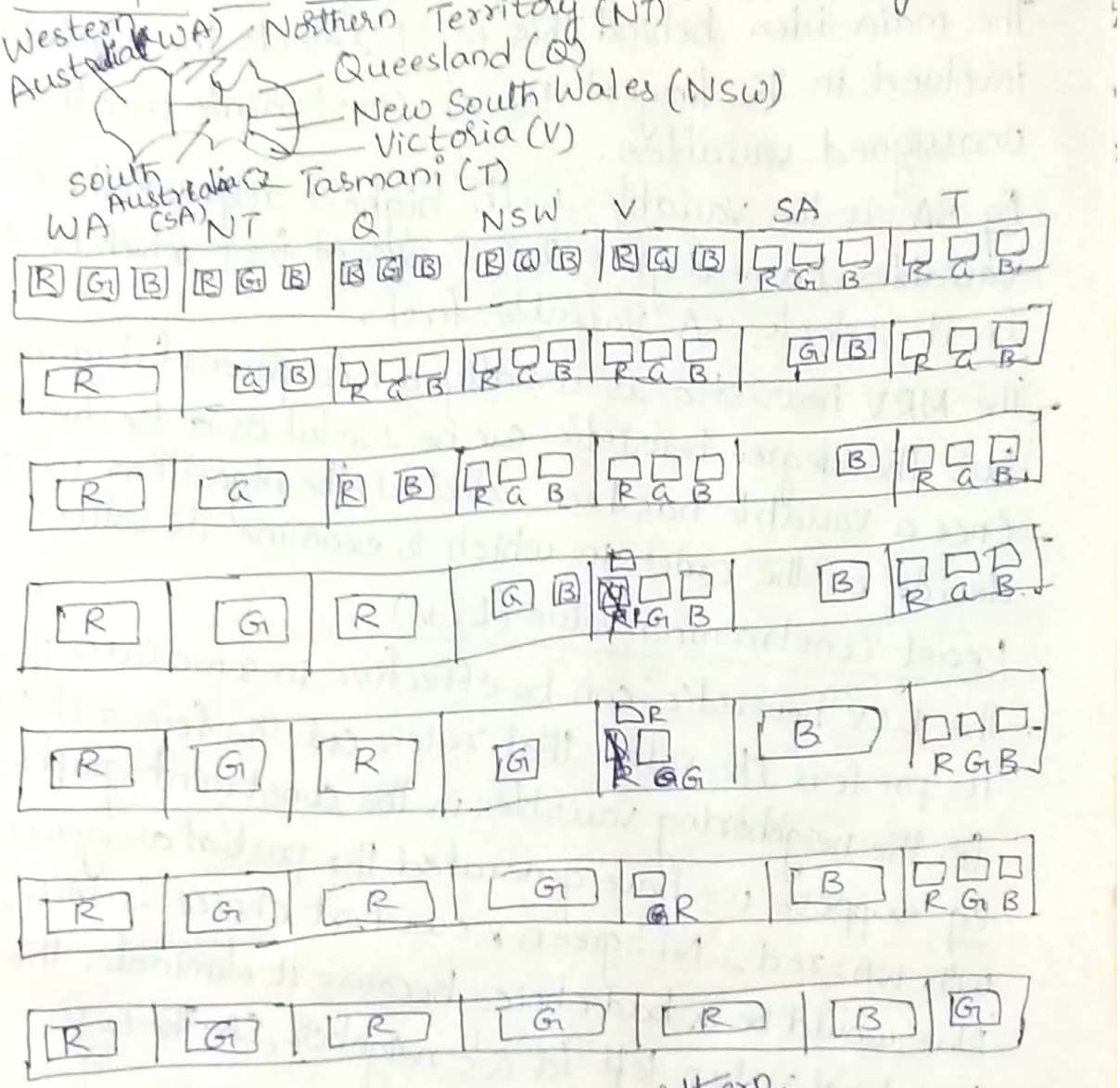
Least Constraining Value (LCV)

- The LCV heuristic can be effective in some cases if it prefers the value that rules out the fewest choices for the neighboring variables in the constraint graph.

Eg suppose we have generated the partial assignment with WA = red & NT = green & our next choice is for Q, Blue would be a bad choice because it eliminates the last legal value left for Q's neighbor, SA. The LCV therefore prefers red to blue.

- In general, the heuristic is trying to leave the max flexibility for subsequent variable assignments.

Example for Forward Search – Map Colding Problem



Constraint Propagation

- Constraint Propagation means using the constraints to reduce the no of legal values for a variable, which can reduce the legal values for another variable & so on
- Constraint propagation may be intertwined with the search or it may be done as a preprocessing step, before the search starts.
- In CSP, constraint propagation can also be made while searching.

Local Consistency

- Local consistency is a property in CSP that characterizes the level of constraint propagation achieved during the search process.
- It refers to the extent to which the constraints in a CSP have been enforced among neighboring variables.

Types

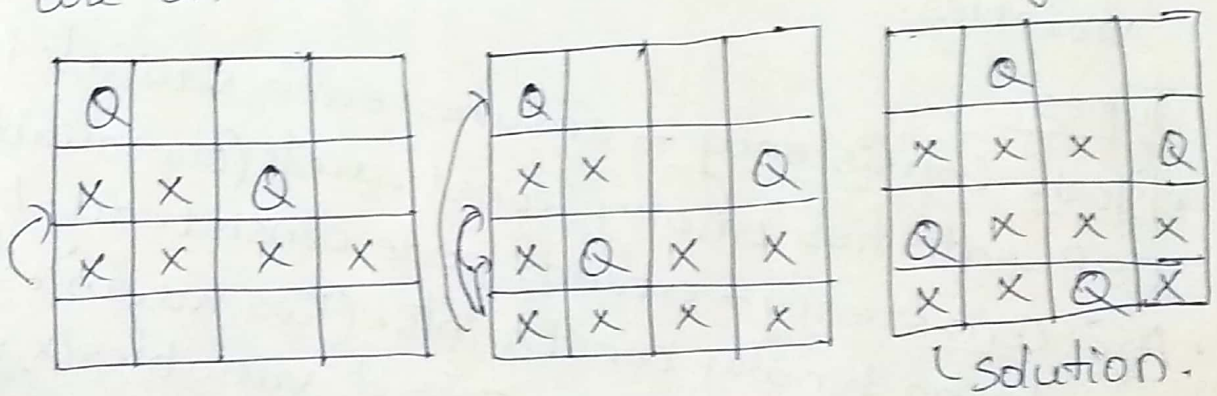
- **Node Consistency** – ensures each variable in CSP satisfies unary constraints (one variable)
- **Arc Consistency** – extends node consistency by considering binary constraints. (Two variables)
  - It ensures that for every pair of variables (x, y) & for every value in the domain of x, there is atleast one value in the domain of y that satisfies the binary constraint. [Not Satisfies – remove value from domain of x]
- **Path Consistency** – It enforces constraints over longer paths in the constraint graph. It checks for consistent values along chains of variables connected by binary constraints.

Backward / Backtracking Search

- Backtracking is a fundamental algorithmic technique for solving CSP
- It systematically explores the search space of possible assignments to variables while using constraints to prune branches unlikely to lead to a solution.
- DFS is used for choosing values in one variable at a time & backwards when a variable has no legal values left to assign.
- DFS for CSP with single-variable assignment is called backtracking search.

Example – 4 Queens Problem

In this problems we have to place 4 Queens on a 4×4 chessboard such that no 2 queens are on the same row, column & diagonal.



Solution.

(b) BackTracking steps

(1) Initialization - start with an initial assignment of values to variables. If some variables are already assigned, this could be an empty or partial assignment.

(2) Select Unassigned Variable - Choose an unassigned variable from the variables that still need to be assigned a value.
- The choice of a variable can be based on various heuristics like the most constrained variable & the variable with fewest legal values.

(3) Order Domain Values - Order the selected variable's domain values. The order can be based on heuristics, like the least contraining value that rules out the fewest choices for other variables.

(4) Value Assignment - For each value in the domain of the selected variable:
(i) assign the value to the variable
(ii) check if the assignment violates any constraints with the already assigned variables. If a constraint is violated backtrack to step-2

(5) Constraint Propagation - After assigning a value, apply constraint propagation techniques. This could involve revising the domains of other variables based on the newly assigned variable & the constraints. Common techniques include arc consistency & domain reduction.

(6) Recursive step & Backtrack -

(a) If no inconsistency is found, after assigning a value & applying constraint propagation then proceed recursively to next variable by repeating steps 2 to 5.

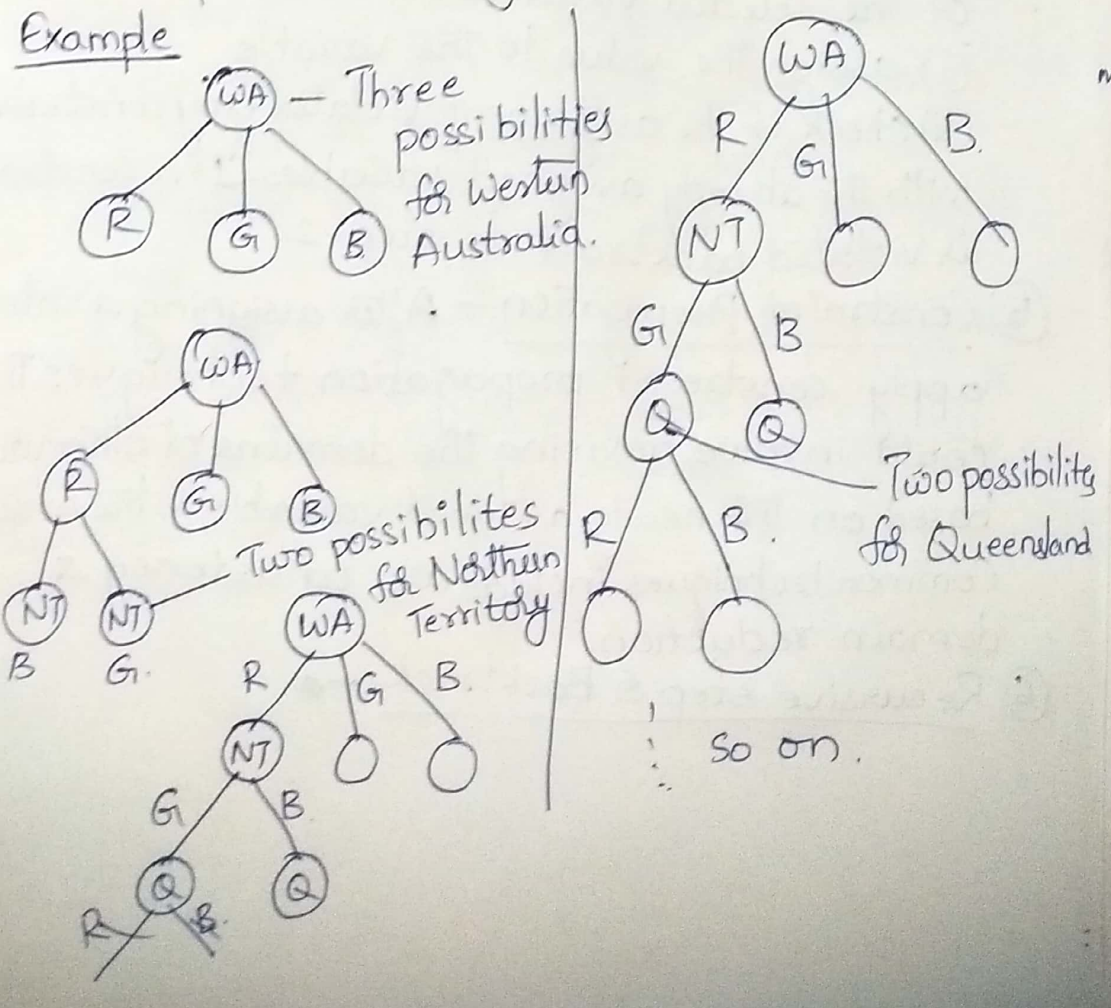
(b) If a variable's domain becomes empty due to the assignment (value), backtrack to the previous variable & undo the assignment.

(7) Solution Found – A solution has been found if all variables are assigned values & all constraints are satisfied. Return the assignment.

(8) Backtrack – If the search reaches dead-end (no values), backtrack to the previous variable undo its assignment & continue the search process.

(9) Termination – Continue Until a solution is found & all possible assignments have been explored.

Example



Knowledge Representation

- We, humans are good at reasoning, understanding, analyzing & interpreting what we are seeing in our daily life. We know how to react & respond to the situation we are going Through. But how to make the machines perform similar actions?
- Here comes the knowledge representation in AI that equips the machines to perform understanding & interpreting the queries of the real world.
- Knowledge Representation in AI describes the representation of knowledge. Basically, it is a study of how the beliefs, intentions & judgements of an intelligent agent can be expressed suitably for automated reasoning.
- One of the primary purposes of knowledge Representation includes modeling intelligent behavior for an agent.
- Knowledge Representation & Reasoning (KRR) represents info from the real world for a computer to understand & then utilize this knowledge to solve complex real-life problems like communicating with human beings in natural language.
- Knowledge Representation in AI is not just about storing data in a database, it allows a machine to learn from that knowledge & behave intelligently like a human being.

Different Types of Knowledge

- Declarative Knowledge - It includes concepts, facts & objects & expressed in a declarative sentence.
- Structural Knowledge - It is a basic problem-solving knowledge that describes the relationship b/w concepts & objects.
- Procedural Knowledge - This is responsible for knowing how to do something & includes rules, strategies, procedures etc
- Meta Knowledge - It defines knowledge about other types of knowledge.
- Heuristic Knowledge - This represents some expert knowledge in the field & subject.

Techniques of knowledge Representation

Three major techniques of knowledge representation:

① Logical Representation

- It is the most basic form of representing knowledge to machines where a well-defined syntax with proper rules is used.
- This syntax needs to have no ambiguity in its meaning & must deal with propositions
- Thus, this logical form of presentation acts as communication rules & is why it can be best used when representing facts to a machine.

- Logical representation is a lang with some definite rules which deals with propositions & has no ambiguity in representation.
- It represents a conclusion based on various conditions & lays down some imp communication rules.
- Logical representation can be categorized into mainly two logics:
  - Propositional Logic
  - First Order Logic.
- Propositional Logic, also known as propositional calculus, or stmt logic, is a formal system of logic that deals with the relationships b/w propositions, which are stmts that are either true or false.
- It is based on the Boolean system, which means that propositions are evaluated as either true or false. In propositional logic, propositions are combined using logical connectives such as 'AND', 'OR' & 'NOT' & the resulting compound propositions can also be evaluated as true or false based on the truth values of their component propositions.
- First Order Logic also known as first order predicate logic calculus or first order logic with identity, is an extension of propositional logic that allows for the representation of more complex relationships b/w objects. In FOL, propositions are constructed using predicates, which are stmts that describe properties & relations b/w objects & quantifiers which specify the scope of the variables in the proposition.

Advantages

- It helps to perform logical reasoning
- This representation is the basis for the programming lang

Disadvantages

- It have some restrictions & are challenging to work with.
- This technique may not be very natural & inference may not be every efficient.

Propositional Logic

- It represents real world facts as logical proposition written as Well formed Formulas
- Simple to deal with
- A decision procedure exists for it.

Facts	Propositional Logic
It is raining	— Raining
It is Sunny	— Sunny
It is Windy	— Windy

If it is raining, then it is not Sunny — Raining → ¬Sunny

Summarized Table for Propositional Logic Connectives

Connective Symbols	Word	Technical Term	Example
∧	AND	Conjunction	A ∧ B
∨	OR	disjunction	A ∨ B
¬ & ~	NOT	Negation	¬A & ¬B
→	Implies	Implication	A → B
⟺	if and only if	Biconditional	A ⟺ B

Logical Equivalence

(1) Identity – $p \wedge T \equiv p$, $p \vee F \equiv p$, $p \vee T \equiv T$

(2) Domination – $p \vee T \equiv T$, $p \wedge F \equiv F$

(3) Idempotent – $p \wedge p = p$, $p \vee p = p$

(4) Double Negation – $\neg(\neg p) = p$

(5) Commutative – $p \vee q = q \vee p$, $p \wedge q = q \wedge p$

(6) Associative – $(p \vee q) \vee r \equiv p \vee (q \vee r)$, $(p \wedge q) \wedge r = p \wedge (q \wedge r)$

(7) Distributive – $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$

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

(8) De Morgan's Law – $\neg(p \wedge q) = \neg p \vee \neg q$

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

(9) Absorption – $p \vee (p \wedge q) = p$, $p \wedge (p \vee q) = p$

(10) Negation – $p \vee \neg p = T$, $p \wedge \neg p = F$

Limitations

- If you want to represent complicated sentence in natural lang structs, PL is not sufficient.
- We cannot represent relations like ALL, some etc with propositional logic

Eg All the girls are intelligent.
Some apples are sweet.

- There is very limited expressive power in propositional Logic, so we use FOL instead

Eg All men are mortal Mortal Man

- fails to capture the relationship btw individual being a man & that individual being a mortal.

First Order Logic

- First Order Logic is another way of knowledge representation in AI. It is an extension to propositional Logic.
- FOL is sufficiently expressive to represent the natural lang stmts in a concise way.
- First Order Logic is also known as Predicate Logic or First-Order predicate logic. It is a powerful lang that develops info about the objects in a more easy way & can also express the relationship b/w those objects.

Quantifiers

- These are the symbols that permit to determine or identify the range & scope of the variable in the logical expression. There are two types of quantifier
  - Universal Quantifier (for all, everyone, everything - $\forall$)
  - Existential Quantifier (for some, atleast one - $\exists$)
- If x is a variable, then $\forall x$ is read as
  For all x (or) For each x (or) For every x.

Eg All man drink coffee
Let a variable x which refers to man, so all x can be represented as: $\forall x \; man(x) \rightarrow drink(x, coffee)$.

- It will be read as: There are all x where x is a man who drink coffee.
- If x is a variable, then existential quantifier will be $\exists x$ & $\exists(x)$. And it will read as
  There exists a 'x' (or) For some 'x' (or) For atleast one 'x'

Eg Some boys are intelligent
$\exists x : boys(x) \wedge intelligent(x)$

- It will be read as: There are some x where x is a boy who is intelligent.

(1) Marcus was a man — Man(Marcus)

(2) Marcus was a pompeian — Pompeian(Marcus)

(3) All pompeians were Romans — $\forall x$: Pomepian(x) → Roman(x)

(4) Cesear was a ruler — Ruler(Cesear)

(5) All romans are either loyal to cesear or hated him

$\forall x$: Roman(x) → loyalto (x, ceaser) ∨ hated (x, cesear)

(6) Everyone is loyal to someone $\forall x \exists y$: loyalto(x, y).

(7) People only try to assassinate rulers they are not loyal to $\forall x : \forall y$: person(x) ∧ ruler(y) ∧ tryassassinate (x, y) → ¬ loyalto(x, y)

(8) Marcus tried to assassinate cesear — tryassassinate (Marcus, caeser)

Resolution

It is an inference rule which is used in both propositional as well as First order Predicate Logic in different ways. This method is basically used for proving the satisfiability of the problem. In resolution method, we use Proof by Refutation technique to prove the given statement.

- Resolution is used, if there are various statements are given & we need to provide a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the conjunctive Normal Form (CNF) & clausal form.

Clause - Disjunction of literals (an atomic sentence) is called a clause. It is also known as Unit Clause.

CNF - A sentence represented as a conjunction of clauses is said to be CNF or conjunctive Normal Form

- The key idea for the resolution method is to use the Knowledge base & negated goal to obtain NULL clause (which indicates contradiction).

Steps for Resolution

① Conversion of facts into first-Order logic (FOL)
② Convert FOL statements into CNF
③ Negate the statement which needs to prove (proof by contradiction)
④ Draw the resolution graph(unification).

Example

ⓐ John likes all kinds of food
ⓑ Apple & vegetable are food
ⓒ Anything anyone eats & not killed is food
ⓓ Anil eats peanuts & still alive
ⓔ Harry eats everything that Anil eats
Prove by Resolution that:
ⓕ John likes peanuts

Step1 – Conversion of facts into FOL.

ⓐ $\forall x$: food(x) → likes(John, x)
ⓑ food(Apple) ∧ food(Vegetables)
ⓒ $\forall x \forall y$: eats(x,y) ∧ ¬killed(x) → food(y).
ⓓ eats(Anil, Peanuts) ∧ alive(Anil)
ⓔ $\forall x$: eats(Anil, x) → eats(Harry, x)
ⓕ $\forall x$: ¬killed(x) → alive(x)] added predicates
ⓖ $\forall x$: alive(x) → ¬killed(x)]
ⓗ likes(John, Peanuts)

Step 2 – Conversion of FOL into CNF

This step is required as CNF form makes easier for resolution proofs in FOL resolution.

(a) Eliminate all implication (→) and rewrite.

(a) $\forall x \neg Food(x) \vee likes(John, x)$ $\quad [P \rightarrow Q \Rightarrow \neg P \vee Q]$

(b) $food(Apple) \wedge food(vegetables)$

(c) $\forall x \forall y \neg [eats(x, y) \wedge \neg killed(x)] \vee food(y)$

(d) $eats(Anil, Peanuts) \wedge alive(Anil)$

(e) $\forall x \neg eats(Anil, x) \vee eats(Harry, x)$

(f) $\forall x \neg (\neg killed(x)) \vee alive(x)$

(g) $\forall x \neg alive(x) \vee \neg killed(x)$

(h) $likes(John, Peanuts)$

(b) Move negation (¬) inwards & rewrite.

(a) $\forall x \neg Food(x) \vee likes(John, x)$

(b) $food(Apple) \wedge food(vegetables)$

(c) $\forall x \forall y \neg eats(x, y) \vee killed(x) \vee food(y)$

(d) $eats(Anil, Peanuts) \wedge alive(Anil)$

(e) $\forall x \neg eats(Anil, x) \vee eats(Harry, x)$

(f) $\forall x \; killed(x) \vee alive(x)$

(g) $\forall x \neg alive(x) \vee \neg killed(x)$

(h) $likes(John, Peanuts)$

(c) Rename variables & standardize variables.

(a) $\forall x \neg food(x) \vee likes(John, x)$

(b) $food(Apple) \wedge food(Vegetable)$

(c) $\forall y \forall z \neg eats(y,z) \vee killed(y) \vee food(z)$

(d) $eats(Anil, Peanuts) \wedge alive(Anil)$

(e) $\forall w \neg eats(Anil, w) \vee eats(Harry, w)$

(f) $\forall g \neg killed(g) \vee alive(g)$

(g) $\forall k \neg alive(k) \vee \neg killed(k)$

(h) $likes(John, Peanuts)$

(d) Eliminate existential quantifier by elimination

In this step, we will eliminate existential quantifier ∃ & this process is known as skolemization. But in this example no existential quantifier so all the statements will remain same in this step.

(e) Drop Universal quantifiers

(a) $\neg food(x) \vee likes(John, x)$

(b) $food(Apple)$ (c) $food(Vegetables)$

(d) $\neg eats(y,z) \vee killed(y) \vee food(z)$

(e) $eats(Anil, Peanuts)$ (f) $alive(Anil)$

(g) $\neg eats(Anil) \vee eats(Harry, w)$

(h) $killed(g) \vee alive(g)$

(i) $\neg alive(k) \vee \neg killed(k)$

(j) $likes(John, Peanuts)$

(f) Distribute conjunction ∧ over disjunction ¬

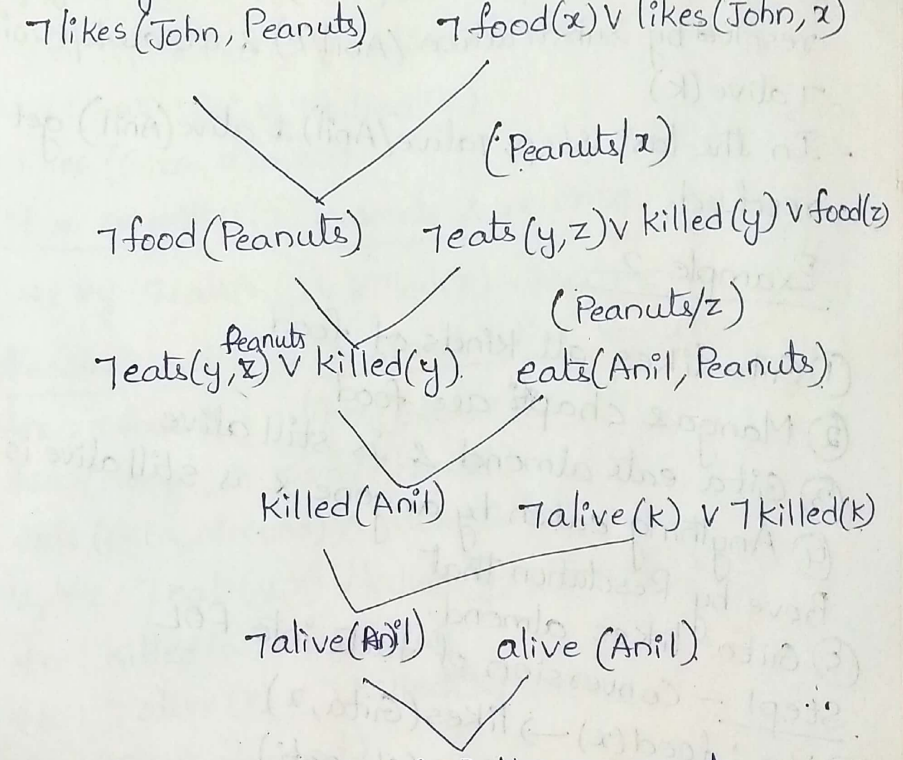
- This step will not make any change in this problem.

Step3 - Negate the statement to be proved

In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)

Step4 - Draw Resolution graph

Now in this step we will solve the problem by resolution tree using substitution.



Explanation

- In the first step of resolution graph, ¬likes(John, Peanuts) & likes(John, x) get resolved (canceled) by substitution of (Peanuts/x) & we are left with ¬food(Peanuts).

- In the second step, ¬food(Peanuts) & food(z) get resolved (canceled) by substitution of (Peanuts/z) & we are left with ¬eats(y, Peanuts) ∨ killed(y)
- In the third step, ¬eats(y, Peanuts) & eats(Anil, Peanuts) get resolved (canceled) by substitution (Anil/y) & we are left with killed(Anil)
- In the fourth step, killed(Anil) & ¬killed(k) get resolved by substitution (Anil/k) & we are left with ¬alive(k)
- In the last step; ¬alive(Anil) & alive(Anil) get resolved.

Example-2

① Gita likes all kinds of food
② Mango & chapti are food
③ Gita eats almond & is still alive
④ Anything eaten by anyone & is still alive is Food.
Prove by Resolution that
⑤ Gita likes almond.

Step1 – Conversion of facts into FOL

ⓐ ∀x : food(x) → likes(Gita, x)
ⓑ food(Mango) ∧ food(Chapati)
ⓒ ∀x ∀y : eats(x, y) ∧ ¬killed(x) → food(y)
ⓓ eats(Gita, almond) ∧ alive(Gita)
ⓔ ∀x : ¬killed(x) → alive(x)
ⓕ ∀x : alive(x) → ¬killed(x)
ⓖ likes(Gita, almond)

} added predicates

Step2 – Conversion of FOL into CNF

(a) Eliminate all implications (→) and rewrite [P→Q = ¬PVQ]

(a) $\forall x : \neg food(x) \vee likes(Gita, x)$

(b) $food(Mango) \wedge food(Chapati)$.

(c) $eats(Gita, almond) \wedge alive(Gita)$.

(d) $\forall x \forall y : \neg [eats(x,y) \wedge \neg killed(x)] \vee food(y)$.

(e) $\forall x : \neg(\neg killed(x)) \vee alive(x)$.

(f) $\forall x : \neg alive(x) \vee \neg killed(x)$.

(g) $likes(Gita, almond)$

(b) Move negation (¬) inwards & rewrite

(d) $\forall x \forall y : \neg eats(x,y) \vee killed(x) \vee food(y)$.

(c) Rename variables or standardize variables.

(a) $\forall x : \neg food(x) \vee likes(Gita, x)$.

(b) $food(Mango) \wedge food(Chapati)$

(c) $eats(Gita, almond) \wedge alive(Gita)$

(d) $\forall y \forall z : \neg eats(y,z) \vee killed(y) \vee food(z)$

(e) $\forall w : killed(w) \vee alive(w)$.

(f) $\forall k : \neg alive(K) \vee \neg killed(K)$

(g) $likes(Gita, almond)$

(d) Eliminate existential quantifiers

No existential quantifiers present in this example

(e) Drop Universal quantifiers

(a) $\neg food(x) \vee likes(Gita, x)$

(b) food (Mango) (c) food (Chapati)
(d) eats (Gita, almond) (e) alive (Gita)
(f) ¬eats(y,z) ∨ killed(y) ∨ killed(z)
(g) ~~¬x~~ killed(w) ∨ alive(w)
(h) ~~¬k~~ ¬alive(k) ∨ ¬killed(k)
(i) likes (Gita, almond)
(j) Distribute conjunction ∧ over Disjunction ∨

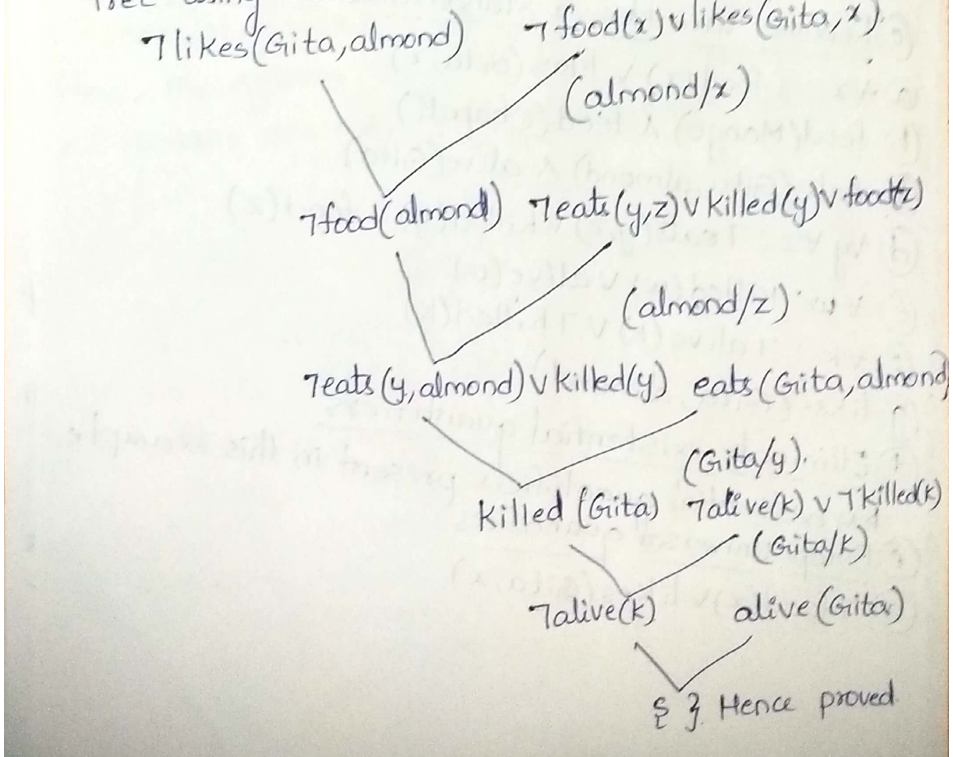
No changes

Step3 - Negate the statement to be proved.

¬ likes (Gita, almond)

Step4 - Draw the Resolution graph.

Now in this step we will solve the problem by resolution reduction tree using substitution



∴ Hence proved

Lifting

It is a process of moving from a specific (ground) level of reasoning to a more abstract (higher) level. It involves generalizing specific instances to form broader concepts or rules that can be applied to other instances.

Example

Consider a scenario where a system knows the following specific facts:

Human (Socrates) Human (Plato) Human (Aristole)

And it also knows:

Mortal (Socrates) Mortal (Plato) Mortal (Aristole)

From these, the system might "lift" the knowledge into a general rule:

$$\forall x \, (\text{Human}(x) \rightarrow \text{Mortal}(x))$$

Here, the system has generalized the concept that all humans are mortal based on specific examples.

Reasoning

It is the process of deriving new knowledge from existing knowledge. It involves applying logical rules to a set of facts to infer conclusions & answer queries.

Types of Reasoning.

① Deductive Reasoning - It uses available facts, info & knowledge to draw valid conclusions.

- It is referred to as top-down reasoning & contradictory to inductive reasoning.
- In deductive reasoning, the truth of the premises guarantees the truth of the conclusion.

Eg Premise-1 : All the humans eats veggies
Premise-2 : Suresh is human.
Conclusion : Suresh eats veggies.

② Inductive Reasoning - It is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalization. It starts with a series of facts & data & researches to a general statement & conclusion.

- In inductive reasoning, we use historical data & various premises to generate a generic rule, for which premises support the conclusion.

Example

Premise - All the pigeons we have seen in the zoo are white

Conclusion - Therefore, we can expect all the pigeons to be white.

(3) Abductive Reasoning - It is a form of logical reasoning which starts with single & multiple observations then seeks to find the most likely explanation & conclusion for the observation.

- It is an extension of deductive reasoning, but in deductive reasoning, the premises do not guarantee the conclusion.

Example

Implication : Cricket ground is wet if it is raining

Axion — Cricket ground is wet

Conclusion - It is raining

(4) Common Senson Reasoning - It is an informal form of reasoning, which can be gained through experiences.

- It simulates the human ability to make presumptions about events which occurs on every day.

- It relies on good judgment rather than exact logic & operates on heuristic knowledge & heuristic rules

Example

(4) One person can be at one place at a time

(5) If I put my hand in a fire then it will burn

(6) Montonic Reasoning - In this, once the conclusion is taken, then it will remain the same even if we add some other info to existing info in our knowledge base.

- To solve these problems we can drive the valid conclusion from the avaliable facts only & it will not be affected by new facts.

Inferencing in AI

Inferencing is a fundamental process in AI where machines make logical deductions & draw conclusion from available information.

- It is the core of intelligent decision-making & problem-solving in AI systems.
- The inference engine is the component of the intelligent system in AI which applies logical rules to the knowledge base to infer new information from known facts. The first inference engine was part of the expert system.
- Inference engine commonly proceeds in two modes, which are:

(1) Forward

- It is a crucial constraint propagation alg used in the field of AI, particularly in the context of CSP.
- It is a powerful technique that helps to efficiently prune the search space & reduce the no of backtracking steps required to find a sol. The fundamental idea behind forward checking is to detect & eliminate infeasible variable assignments as early as possible in the search process, thereby improving the overall performance & efficiency of the problem-solving algorithm.

Principles

- **Early elimination of infeasible Assignments** — Primary principle is to detect & eliminate infeasible variable assignments as early as possible in the search process.
- **Constraint Propagation** — It actively propagates the consequences of each variable assignment throughout the remaining unassigned variables. This process of constraint propagation ensures that the impact of a decision is felt across the entire problem domain, allowing the alg to make informed choices & avoid dead ends.
- **Efficient Backtracking** — when it identifies an infeasible assignment, it triggers a backtracking process to explore alternative sols. This backtracking is guided by the principles of forward checking as the alg can efficiently pin point the decision points that lead to the dead-end allowing it to backtrack to the most appropriate location in the search tree.

Properties

- It is also known as data-driven as we reach the goal using available data.
- It is a down-up approach as it moves from bottom to top.

Example

As per the law, it is a crime for an American to sell weapons to hostile nations. Country Nono, an enemy of America, has some missiles & all the missiles were sold to it by Colonel West, who is American.

Prove that 'Colonel West is Criminal'.

- To solve this problem, first we will convert all the above facts into FOL & then we will use a fdward checking alg to reach the goal.

• Facts conversion to FOL.

- It is a crime for an American to sell weapons to hostile nations:

  $American(x) \wedge weapon(y) \wedge sell(x, y, z) \wedge hostile(z) \Rightarrow criminal(x)$

- Nono has some missiles.

  $\exists x\ Owns(Nono, x) \wedge Missile(x)$ - transformed into two definite clauses.

  $Owns(Nono, M_1) \wedge Missile(M_1)$

- All of its missiles were sold to it by Colonel West

  $Missiles(x) \wedge Owns(Nono, x) \Rightarrow sells(West, x, Nono)$

- Missiles are weapons $Missile(x) \rightarrow weapon(x)$

- An enemy of America counts as 'hostile'

  $Enemy(x, America) \rightarrow Hostile(x)$

- West is American $American(West)$

- The country Nono is an enemy of America

  $enemy(Nono, America)$

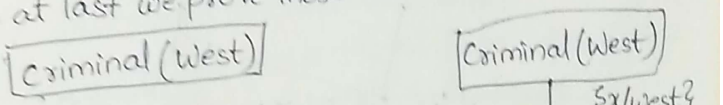
- It is a process of making a conclusion based on known facts or data, by starting from the initial state to the goal state.
- It is commonly used in expert system such as CLIPS, business & production rule systems.

Backward Chaining:

It is also known as a backward deduction or backward reasoning method when using an inference engine. A backward chaining alg is a form of reasoning, which starts with the goal & works backward, chaining through rules to find known facts that support the goal.

Eg - we will start with the goal predicate Criminal(west) & then infer further rules.

Step-1
consider the goal fact & from this we will infer other facts & at last we prove those facts true.

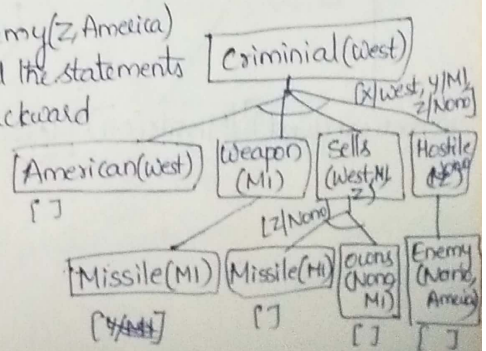


Step2
We will infer other facts from the goal fact which satisfies the rules

step3
We will further extract the fact Missile(M1) which infer from Weapon(M1)

Step4
We infer facts Missile( ) & Owns(M1,z) from Sells(West,M1,z)

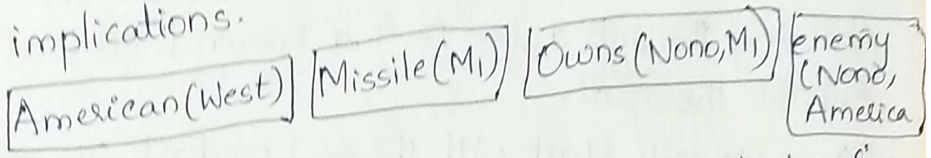
step5
We can infer the fact enemy(z,America) from Hostile(z). Hence all the statements are proved True using backward chaining.



Forward checking Proof

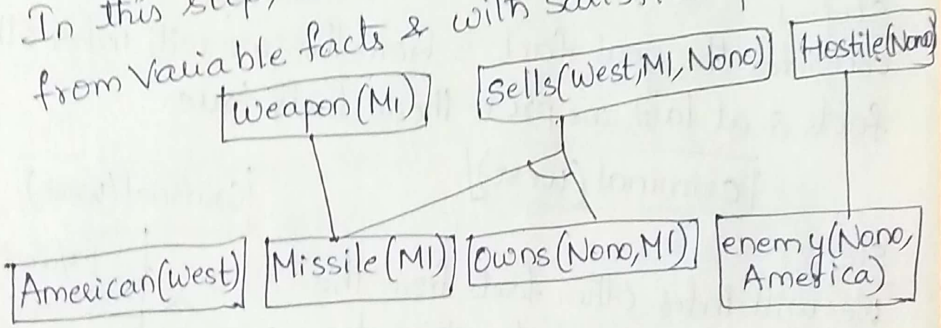
Step-1

In the first step, we will start with the facts & will choose the sentences which do not have implications.



Step 2

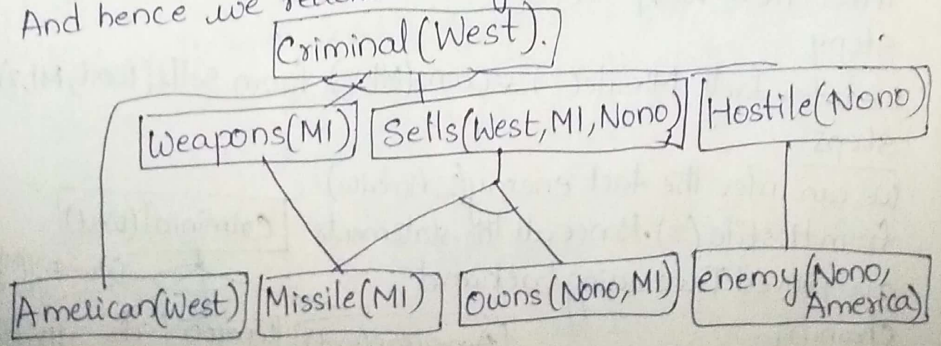
In this step, we will see those facts which infer from Vaiable facts & with satisfied premises.



Step3

At step 3, as we can check Rule(1) is satisfied with the substitution ( ).

And hence we reached the goal.



Probability Theory

Probability is defined as the chance of happening or occurrence of an event. Generally, the possibility of analyzing the occurrence of any event concerning previous data is called probability. For ex, if a fair coin is tossed, what is the chance that it lands on the head? These types of questions are answered under probability.

- Probability measures the likelihood of an event's occurrence. In situations where the outcome of an event is uncertain, we discuss the probability of specific outcomes to understand their chances of happening. The study of events influenced by probability falls under the domain of statistics.
- Probability theory studies random events & tells us about their occurrence. The two main approaches for studying probability theory are:

(1) Theoretical Probability

It deals with assumptions to avoid unfeasible & expensive repetition of experiments.

- The theoretical probability for an event A can be calculated as follows:

$$P(A) = \frac{\text{No of favorable to A}}{\text{Total no of possible outcomes}}$$

Note - Here, we assume the outcomes of an event are equally likely.

For eg tossing a coin - 2 outcomes : Head & Tail.
The probability of occurrence of Head on tossing a coin is $P(H) = 1/2$.
Similarly, probability of occurrence of Tail on tossing a coin is $P(T) = 1/2$

Experimental Probability:
It is found by performing a series of experiments & observing their outcomes. These random experiments are also known as trails.
The experimental probability for event A can be calculated as $P(E) = \frac{(\text{No of times event A happened})}{(\text{Total no of trails})}$.

Experimental probability = P/n

For eg tossing a coin. If we tossed a coin 10 time & recorded heads for 4 times & a tail 6 times ther the probability of occurrence of head on tossing a coin is $P(H) = 4/10$
Similarly, the probability of occurrence of tail on tossir a coin is $P(T) = 6/10$

Eg Let's take two random dice & roll them random now the probability of getting a total of 10 is calculat
Sol Total possible events that can occur (sample space {(1,1)(1,2)(1,3)..(1,6)... (6,6)}. The total spaces are 36.
Now the required events are {(4,6)(5,5)(6,4)} - adds upto
So, the probability of getting a total of 10 is $3/36 = 1/12$

Uncertainty in AI

Uncertainty plays a significant role in AI & ML. It refers to the lack of complete information & the presence of randomness & variability in data & in the outcomes of AI models.

- Dealing with uncertainty is crucial in AI for making informed decisions, handling noisy data & building robust & reliable systems.

Types

- **Aleatroic Uncertainty**

This type of uncertainty arises from inherent randomness & variability in data. It is often associated with observations that are subject to random noise. For eg in computer vision, the position of an object in an image may have aleatroic uncertainty due to variations in lighting & camera sensor noise.

- **Epistemic Uncertainty**

It is related to the lack of knowledge & information. It represents uncertainty that can be reduced with more data & approved model. For eg a ML model may have epistemic uncertainty when trying to make predictions in a data-scarce region.

- **Model Uncertainty**

It encompasses the uncertainty associated with the choice of model architecture & parameters. It reflects the uncertainty in the model's own internal representation of the data. Techniques like Bayesian neural n/w & dropout regularization can help quantify model uncertainty.

Conditional Probability

The probability of occurrence of any event A when another event B already occured in relation to A is known as conditional probability & denoted by P(A/B)

Baye's Theorem

It is used to determine the conditional probability of an event. & used to find the probability of an event, based on the prior knowledge of conditions that might related to that event.

Formula

For any two events A & B, then the formula of Bayes theorm is given by

$$P(A/B) = \frac{P(B/A)\,P(A)}{P(B)}$$

where P(A) & P(B) are the probabilities of event A & B
P(A/B) - probability of event A when event B happens
P(B/A) - probability of event B when event A happens

Example

Same Space for events A & B

A holds	T	T	F	F	T	F	F
B holds	T	F	T	F	T	F	F

P(A) = 4/7 [out of 7 A contains 4 T]

P(B) = 3/7 [out of 7 B Contains 3 T]

$$P(A/B) = \frac{P(A \cap B)}{P(B)} = \frac{2 \text{ (both contain T)}}{3 \text{ (B contains T)}}$$

$$P(B/A) = \frac{P(A \cap B)}{P(A)} = \frac{2 \text{ (both contains T)}}{4 \text{ (A contains T)}}$$

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)} = \frac{2}{4} * \frac{4}{7} \Big/ \frac{3}{7} = \frac{2}{3} \text{ - correct}$$

$$P(B/A) = \frac{P(A/B) * P(B)}{P(A)} = \frac{\frac{2}{3} * \frac{3}{7}}{\frac{4}{7}} = \frac{2}{4} \text{ - correct}$$

Naive - Bayes Classifier

It is a type of supervised learning which contains labelled data and is used for solving classification problems.

- It is used in text classification, data contains high-dimensions (as each word represents one feature in the data).
- It is used in spam filtering, sentiment detection, rating classification etc.
- This model predicts the probability of an instance belongs to a class with a given set of feature (attribute) value. It is a probabilistic classifier.
- The advantage of using this is its speed. It is fast in making predictions with high-dimension data.
- It uses bayes theorem for training & prediction.
- The naive bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value.
- For a given target value of the instance, the probability of observing conjunction $a_1, a_2 \ldots$ an is just the product of the probabilities for the individual attributes $P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$ (attributes / o/p attribute)
- Naive Bayes classifier:

$$V_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$$

(v_j: values in the variable)

Example

- Consider a fictional dataset that describes the weather conditions for playing a game of golf.

	Outlook	Temperature	Humidity	Windy	PlayGolf
1	Rainy	Hot	High	False	No
2	Rainy	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Sunny	Mild	High	False	Yes
5	Sunny	Cool	Normal	False	Yes
6	Sunny	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Rainy	Mild	High	False	No
9	Rainy	Cool	Normal	False	Yes
10	Sunny	Mild	Normal	False	Yes
11	Rainy	Mild	High	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Sunny	Mild	High	False	No

$P(Yes) = 9/14$

$P(No) = 5/14$

PlayGolf — output or class variable.

Outlook

	Yes	No	P(Yes)	P(No)
Sunny	3	2	3/9	2/5
Overcast	4	0	4/9	0/5
Rainy	2	3	2/9	3/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(Yes)	P(No)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(Yes)	P(No)
High	4	3	4/9	4/5
Normal	5	1	5/9	1/5
Total	9	5	100%	100%

Windy

	Yes	No	P(Yes)	P(No)
False	6	3	6/9	3/5
True	3	2	3/9	2/5
Total	9	5	100%	100%

Play Golf		P(Yes/No) / P(Total)
Yes	9	9/14
No	5	5/14
Total	14	100%

- $P(\text{Outlook} = \text{Sunny} / \text{yes}) = \dfrac{P(\text{yes} / \text{outlook} = \text{Sunny})\, P(\text{O} = \text{Sunny})}{P(\text{yes})}$

$$= \frac{\frac{3}{9} * \frac{9}{14}}{\frac{9}{14}} = \frac{3}{9}$$

- $P(\text{Outlook} = \text{Sunny} / \text{No}) = \dfrac{P(\text{No} / \text{outlook} = \text{Sunny})\, P(\text{O} = \text{Sunny})}{P(\text{Yes})}$

$$= \frac{\frac{2}{5} * \frac{5}{14}}{\frac{5}{14}} = \frac{2}{5}$$

- New instance to classify: Humidity
  today = (Sunny, Hot, Normal, False)
  Outlook Temperature windy
- Our task is to predict the target value (yes/no) of the target concept PlayGolf for this new instance

$$V_{NB} = \underset{v_j \in \{Yes, No\}}{\operatorname{argmax}} P(v_j) \cdot \prod_i P(a_i | v_j)$$

$$= \underset{v_j \in \{Yes, No\}}{\operatorname{argmax}} P(v_j) \prod_i P(today | v_j)$$

$$= \underset{v_j \in \{Yes, No\}}{\operatorname{argmax}} P(v_j)\, P(Outlook = Sunny | v_j) * P(Temperature = Hot | v_j) * P(Humidity = Normal | v_j) * P(Windy = false | v_j)$$

$$= \underset{v_j \in \{Yes, No\}}{\operatorname{argmax}} \left[\left(P(Yes) * P(O = Sunny | Yes) * P(T = Hot | Yes) * P(H = Normal | Yes) * P(W = false | Yes) \right) / \left(P(No) * P(O = Sunny | No) * P(T = Hot | No) * P(H = Normal | No) * P(W = false | No) \right) \right]$$

$$= \text{argmax} \left[\left(\frac{9}{14} * \frac{3}{9} * \frac{2}{9} * \frac{5}{9} * \frac{6}{9} \right) / \left(\frac{5}{14} * \frac{3}{5} * \frac{2}{5} * \frac{1}{5} * \frac{3}{5} \right) \right]$$

$$= \text{argmax} \left[\frac{10}{567}, \frac{6}{875} \right] = \text{argmax}(0.01764, 0.00686)$$

highest value

Thus, the naive Bayes classifier assigns the target value PlayGolf = Yes to this new instance, based on the probability estimates learned from the training data.