?- atom_codes (tiger, X)
X = [116, 105, 103, 101, 114]
Yes

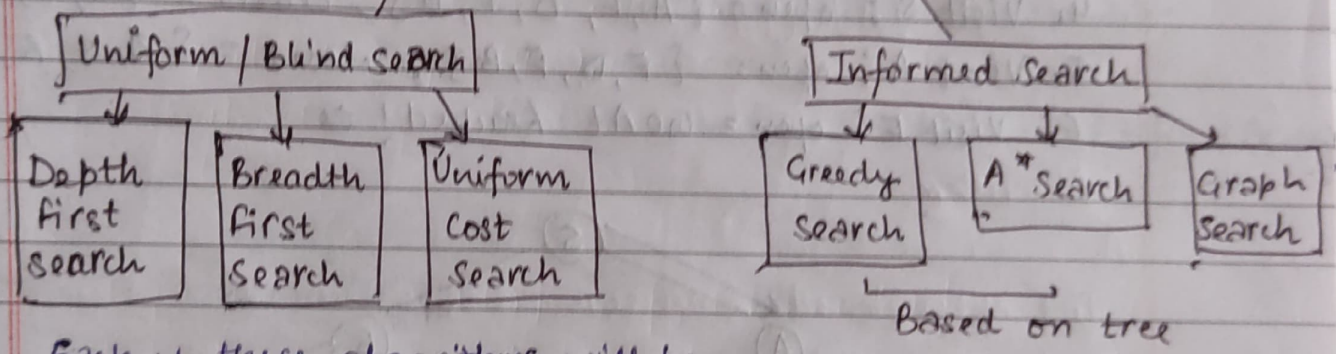
?- atom_codes (tom, A)
A = [116, 111, 109]
Yes

Aug 8

write a program to find cube of a number.

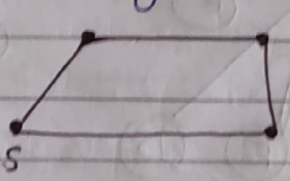
Friday
22/09/2022

Types of search Algorithms




Each of these algorithms will have :

1. A problem graph containing start node S & goal node G.




2. A strategy, describing the manner in which the graph will be traversed to get to G.

3. A fringe, while is a structure used to store all the possible states (nodes) that you can go from the current states.

4. A tree, that results, while traversing to the goal node.

Good Write

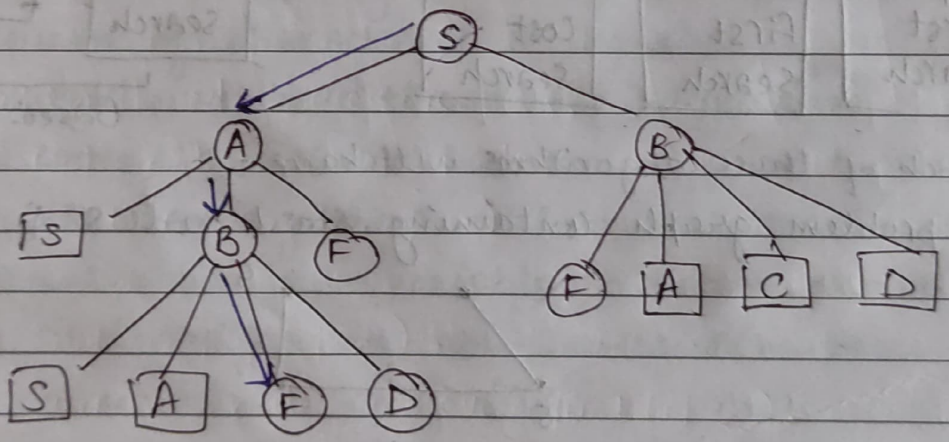
DATE: __/__/____
PAGE: _____

5. A solution plan, while the sequence of nodes from S to G.

## Depth first Search

open = {S} closed = { }

0. Visit S: open = {A, B}, closed = {S}
1. Visit A: open = {S, B, F, B}, closed = {A, S}
2. Visit S: open = {B, F, B}, closed = {S, A, S}
3. Visit B: open = {S, A, F, D, F, B}, closed = {B, S, A, S}
4. Visit S: open = {A, F, D, F, B}, closed = {S, B, S, A, S}
5. Visit A: open = {F, D, F, B}, closed = {A, S, B, S, A, S}
6. Visit F: open = GOAL Reached!




- DFS: complexity

- Time complexity = $O(b^m)$ m is depth of sol$^n$.
- Space complexity = $O(bm)$

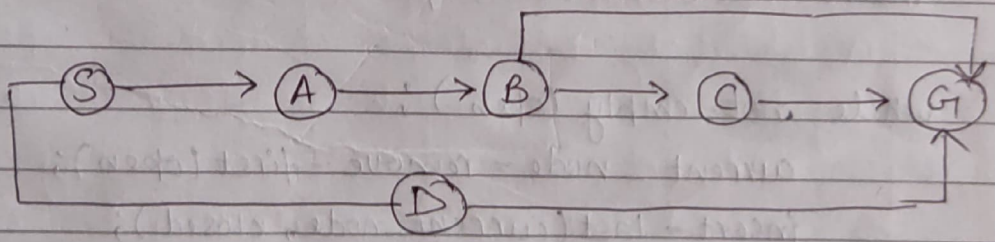
Good Write

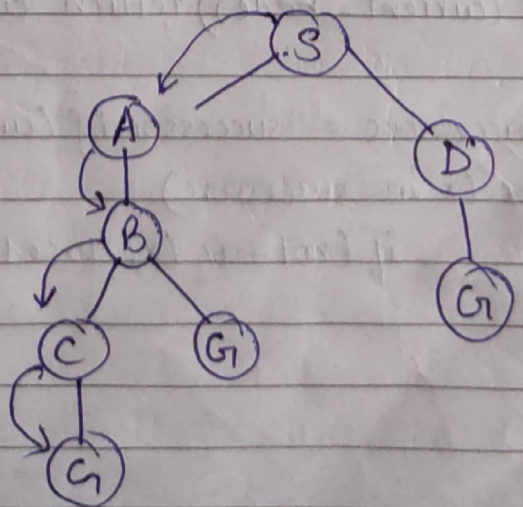
DATE: __/__/____
PAGE: _____

DFS :- Algorithms

```
List open, closed, successors = { };
Node root_node, current_node;
insert - first (root_node, open)

while not - empty (open);
    current_node = remove- first (open);
    insert - first (current_node, closed);
    if (goal (current_node)) return current_node;
    else
        successors = successors of (current_node);
        for (x in successors)
            if (not - in (x, closed)) insert - fixed (x, open);

    endif
endwhile
```

Page 27 DFS




Good Write

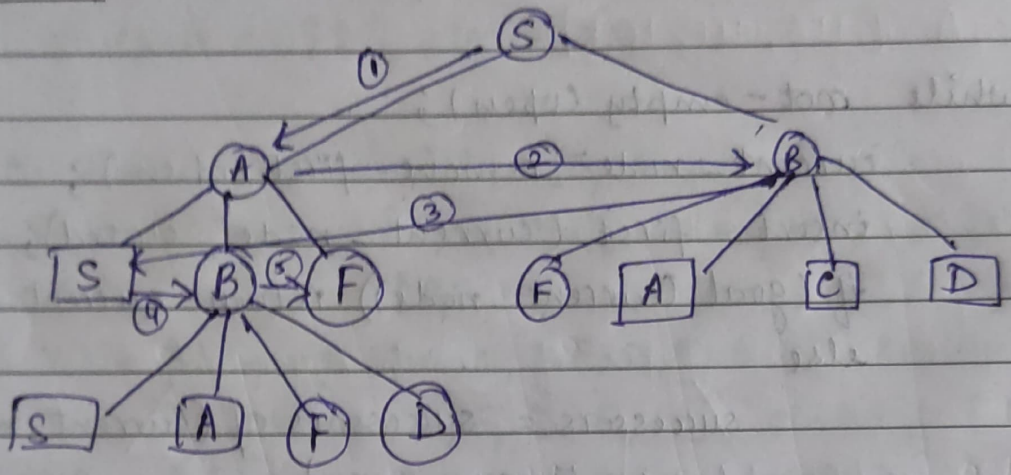
Uninformed

DATE: __/__/____
PAGE: _____

# Breadth first search

## Example & Algorithm

### Example



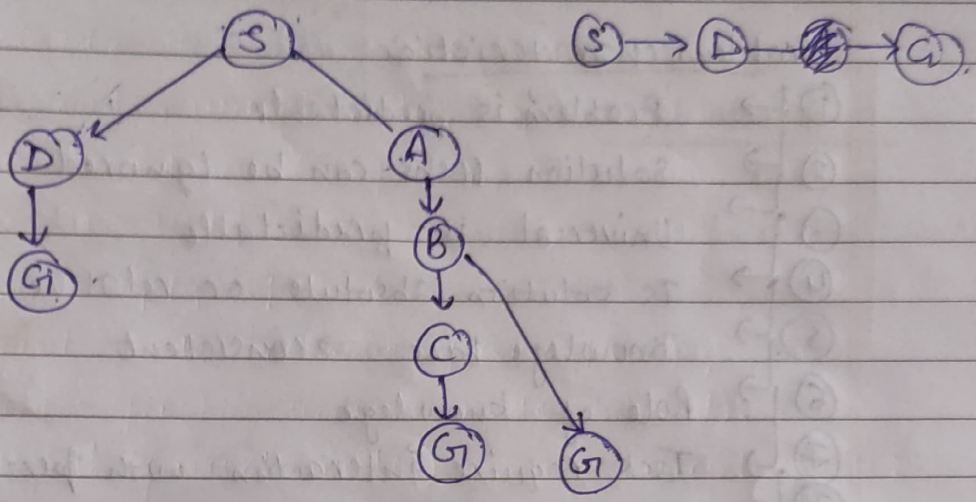

### Algorithm

```
List open, closed, successors = {};
Node root_node, current_node;
insert-last (root_node, open)

while not-empty (open);
        current_node = remove-first (open);
        insert-last (current_node, closed);
        if (goal (current_node)) return current_node;
        else
                successors = successorsOf (current_node);
                for (x in successors)
                        if (not-in (x, closed)) insert-last (x, open);

endif
endWhile
```

Pg 36

Q

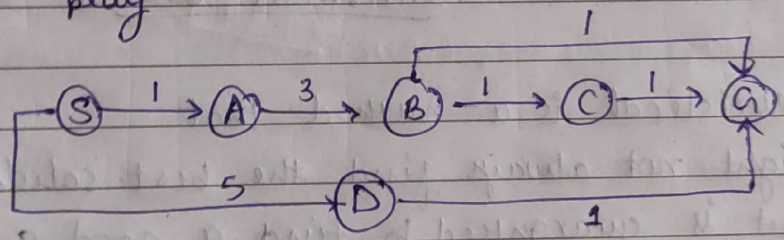
BFS



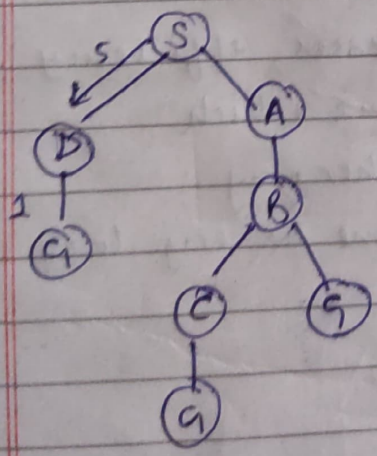

Pg 39

Uniform Cost Search

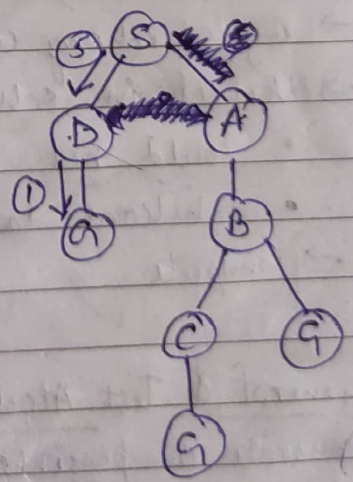
- UCS is different from BFS & DFS because here the costs come into play




DFS (cost = 6)




BFS (cost = 6)



Friday
DATE: 29/09/2023
PAGE:

# Informed Search Algorithms

## Problem characteristics

1. Problem is predictable
2. Solution steps can be ignored
3. Universal is predictable
4. Is solution absolute or relative.
5. Knowlege base → consistent
6. Role of knowlege
7. Task require interaction with person
8. Problem classification

Tuesday
03/10/2023

Pg 44

## Heuristical search

- A heuristical is a method that
  - → might not always find the best solution
  - → but is guaranteed to find a good solution in reasonable time.
  - → By searching completeness it increases efficiency.
  - → Useful in solving tough problems which
    - could not be solved any other way.
    - solutions take an infinite time or very long time.
  - → compute

## General & Test Algorithm Pg 45

- Generate a possible solution which can either be a point in the problem space or a path from the initial state.
- Test to see if this possible solution is a real solution by comparing the state reached with set of goal states.
- If it is a real solution, return. Otherwise repeat from 1.

Good Write

GN → Goal Node

DATE: __/__/____
PAGE: _____

Pg 46

* Hill climbing
  ↳ Drawback → It is not complete unless we introduce backtracking.
  It is not optimal. → Solution find is a local optimum.

Pg 47
56- Pseudo code

Hill Climbing Algorithm / Greedy search Method / Greedy local search
↳ used for optimizing mathematical problems.

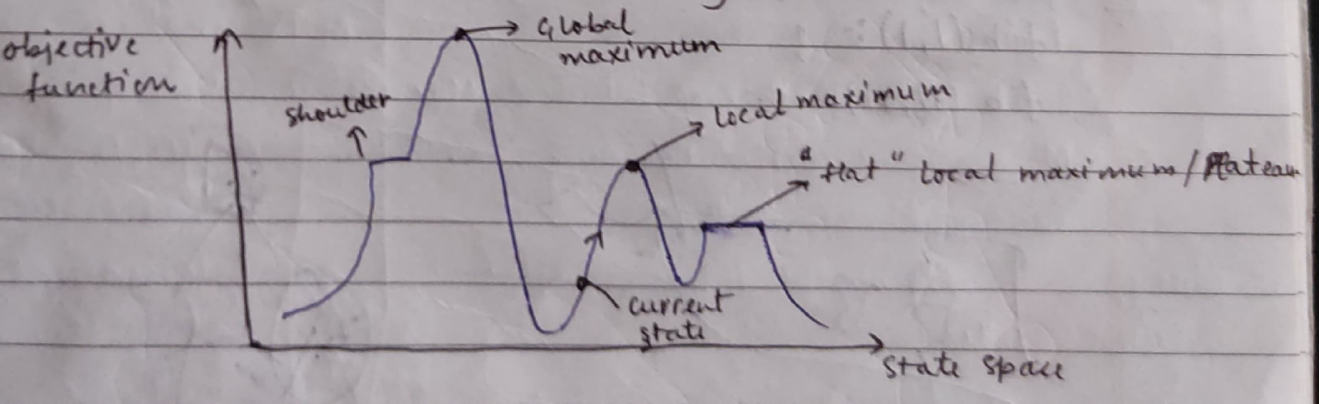
Hill climbing node has 2 components
State / value

Pg 48

Features of Hill climbing
1. Generate & Test variant → produce feedback which helps to decide which direction to move in the search space
2. Greedy approach → Hill climbing search moves in direction which optimizes the cost.
3. No backtracking.

Pg 49

State space Diagram for Hill Climbing




1. Local Maximum: State better than its neighbouring state, but there is also another state which is higher (Global Max.)
2. Global Max:- Best possible state of state space landscape.
   ↳ Highest value of objective function.
3. Current State:- State where agent is currently present.
4. Flat Local Maximum:- All neighbouring state have same value.

Good Write

DATE: __/__/____
PAGE: _____

## Types of Hill-Climbing Algorithm

1. Simple Hill Climbing (SHC)
   - → less optimal sol$^n$ & sol$^n$ is not guaranteed.
   - → less time consuming.
   - → Simplest way to implement a hill climbing algorithm.
   - Evaluates neighbour node at a time & selects the 1st one which optimizes current cost & set it as a current state.

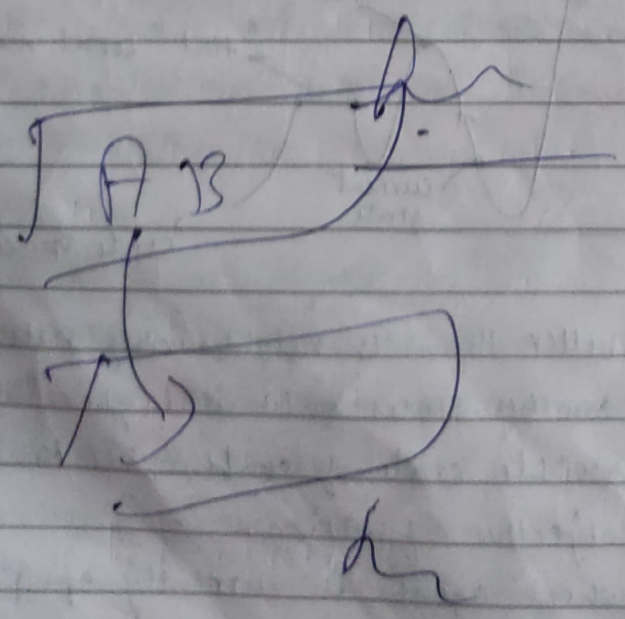
2. Steepest-Ascent Hill-Climbing (SAHC)
   - → Examines all the neighbouring nodes of the current state and selects one neighbour node which is closest to goal node
   - → more time consuming as it searches for multiple neighbours.

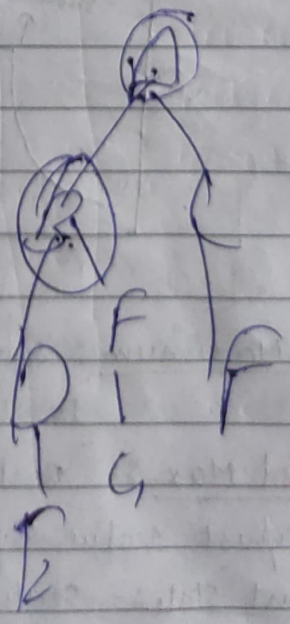
Pg 53 Algorithm for SHC, (SAHC) Pg 54

---

## Fibonacci Series

fib (0,1) :- !.
fib (1,1) :- !.
fib (N,F)



SECTION - A

DATE: __/__/____
PAGE: _____

Ex 1) Fibonacci series

```
fib(0,0):-!.
fib(1,1):-!.
fib(N,F):-
    N>1,
    N1 is N-1,
    N2 is N-2,
    fib(N1,F1),
    fib(N2,F2),
    F is F1+F2.
```

* Calculate factorial

```
factorial(0,1).
factorial(N,M):-
N>0,
Prev is N-1,
factorial(Prev,M1),
M is M1*N.
```

Ex-3 Local variables & Conditional statements

output numbers.

```
eats(plant, grasshopper).
eats(grasshopper, frog).
eats(frog, snake).
eats(snake, hawk).
food_web(X,Y):-
eats(X,Y);
format('A ~w eats ~w', [X,Y]),
numbers:-
write('Enter A:'),
read(A),
nl,
write('Enter B:'),
read(B),
nl,
condition(A,B).
A>B, write('A value is greater than B value');
A=:=B, write('A value is equal to B value');
A<B, write('A value is less than B value').
```

Good Write

DATE: __/__/____
PAGE: _____

Pg no 36

## Best first Search (Greedy Search)

↳ complete but not optimal.

Pg 69 Algorithm 42 Advt & Disad. 43

A* search

$f(n) = g(n) + h(n)$

- $g(n)$: cost to reach node n from start search
- $h(n)$: Cost to reach from node n to goal node
- $f(n)$: Estimated cost of cheapest sol"

→ find shortest path.
→ expand less search tree.
→ back point to check the cover distance.

* 40* Algorithm → Solves AND, OR Problems

Lab (SECTION - B) Ex-2 Facts for statement 3 °C to °F.

11/10/2023 Wednesday

## Unit - II

• Knowledge Representation

Example 1: Family relation (Pg 5)

Facts:
- Sarah is the mother of Tal and Mor
- Mosche is married to Sarah.
- Fanny is the mother of Gal

Query: Is Moche the father of Tal?

Deduction? Yes

Pg 7 Pg 8

- What is knowledge?
- Types of Knowledge
  - → Priori Knowledge (universal, true, not relia...)
  - → Posteriori Knowledge
  - → Procedural Knowledge (knowing how to do)
  - → Declarative Knowledge
    - ↳ Something True / false
  - → Tactic Knowledge
    - ↳ Knowledge not easily expressed by language.

2): Circuit Diagnosis (Pg 6)

Difference

Representation
↳ Symbols standing for in the world or symbolic encoding of propositions.

+ → First Aid

→ Women

Good Write

DATE: / /
PAGE:

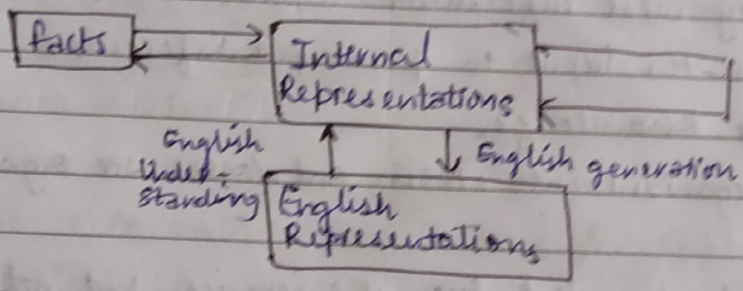
# Reasoning

↳ Manipulation of symbols encoding propositions to produce representations of new propositions.

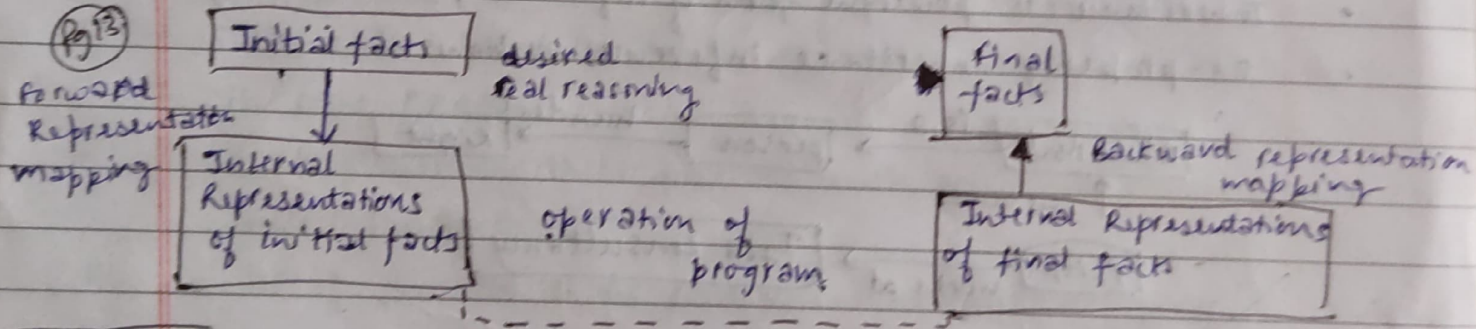
Binary → 1101 + 10 = 1101
(11) (2) (13)

Pg 12

# What is Representation And Mapping ?



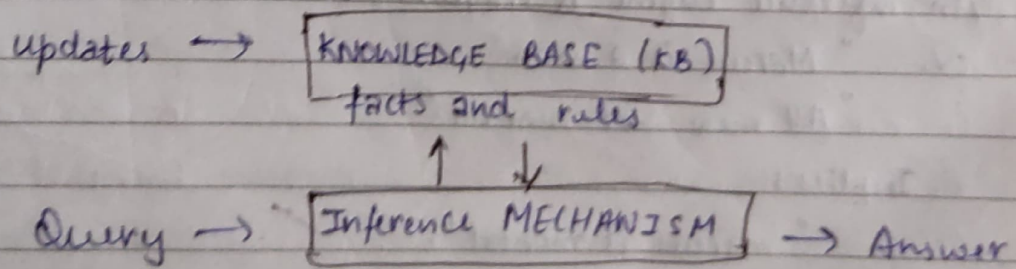

Pg 13




forward representation
↓
left to right
→
Backward
(Opp.)

# Knowledge Base Architecture (KB Architecture)

Pg 14




## Representation & Mapping

- Spot is a dog.
- Every dog has a tail.

spot has a tail.

dog(spot)
∀x : dog(x) → hastail(x)
~~hastail(x)~~
hastail(spot)

Good Write

DATE: __/__/____
PAGE: _____

# Knowledge base issues

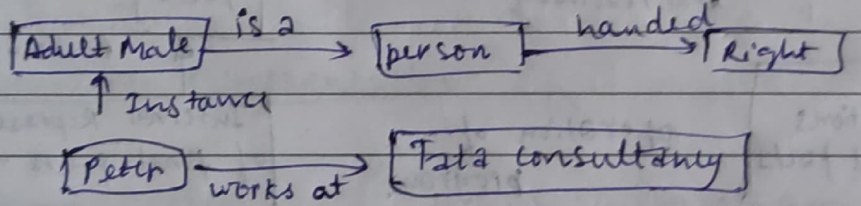
① Representation language: expressive is it? should be uniform universal to avoid it symbol.

② Inference procedure:-
- is it sound? conclusions rationally follows from the facts & rules.
- is it complete? " " " from the KB, can I do ded it?
- is it efficient? Does it take time polynomial in the no of facts & rules

# Approach to KR

① Simple ~~rational~~ relational knowledge
- Provides very west inferential capabilities.
- May serve as the input to powerful inference.

② Inheritable knowledge
- Object are organized into classes and classes are organized in a generalizati hierarchy.
- Inheritance is powerful form of inference, but not adequate.
- ex- property inheritance inference mechanism.




③ Inferential knowledge:
- Facts represented in a logical form, which facilitates reasoning
- An inference (reason) engine is required.

ex. "Marcus is a man"
"All men are mortal."
Implies:
"Marcus is mortal".

④ Procedural knowledge
- how to make it rather than what it is.

# Issues in KR

1. Important Attributes: Isa (related to) and Instance (path) attributes.
2. Relationship among attributes: inverses, existence in a Isa hierarchy, Good Write Single-valued attributes, techniques for reasoning about values.

First order logic = Predicates  verb ( sub , obj )

Axioms : Relationship b/w statements

Sub → agent

DATE : __/__/____
PAGE : ____

3. Choosing the Granularity
   Ex - "John spotted sue":
   spotted ( John , sue )
4. Representing set of objects:
5. Finding the right structure as needed.
   word "fly" have multiply meanings
   1. "John flew to new york." 2. "John flew in a rage," 3. "John flew a kite"

Domain Model / ontology  2. Domain theory
# Representation language
(pg 27) # KB Inference procedure
# Implementation

12/10/2023
Thursday

# A sentence is valid if it is true for any truth assignment. (A V ~A).
# Satisfiable if there exists a truth assignment that make it true (A ∧ B).
# unsatisfiable if there exist no truth assignment that makes it true ( A ∧ ~A)
# model of sentence is an interpretation that satisfies the sentence.

# Proposition • sentence which is true or false.
Proposition symbols / variables : P, Q, S
# Propositional logic ∧, ∨, ⇒, ⇔, ~ / ¬

# Properties of sentence
→ satisfyability → Contradiction → Validity
# Ex - Japan is capital of India. Delhi is the capital of India.

(pg 28) # laws of Algebra of Propositions

| | |
|---|---|
| • Idempotent<br>P ∨ P ≡ P  P ∧ P ≡ P | • Commutative :<br>P ∨ q ≡ q ∨ p  p ∧ q = q ∧ p |
| • Complement<br>P ∨ ~p ≡ T  P ∧ ~p ≡ F | • Double Negation<br>~(~P) ≡ P |

Good Write

DATE: __/__/____
PAGE: _____

- **Associative**
  $p \vee (q \vee r) \equiv (p \vee q) \vee r$
  $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$

- **Distributive:**
  $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
  $p \wedge (q \vee r) \equiv (p \vee q) \vee (p \wedge r)$

- **Absorbtion**
  $p \vee (p \wedge q) \equiv p \wedge (p \vee q)$

- **Identity**
  $p \vee T \equiv T$, $p \vee F \equiv p$, $p \wedge T \equiv p$
  $p \wedge F \equiv F$

- **De Morgan's**
  $\sim(p \vee q) \equiv \sim p \wedge \sim q$
  $\sim(p \wedge q) \equiv \sim p \vee \sim q$

- **Equivalence of contrapositive**
  $p \rightarrow q \equiv \sim p \vee q$
  $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

# **Inference rules** :- logical inference → used to create new sentences that logically follow from given set of predicate calculus sentence (KB)

# **Sound rules of inference** – A rule is sound if its conclusion is true whenever the premise is true.

User provides

Constant Symbols: Green, 3.5, Mary

Function Symbols: father of(Mary) = John

Predicate symbols:
- greater (5,3)
- green (grass)

Tuesday 17/10/2023

(Pg 60) FOL illustrated models for FOL (to Pg 72)

1. Crown is a king & Crown is a person. Crown (King, Person) ~~Crown (King)~~
2. Richard and John both are siblings. Siblings (John) ⇔ siblings (John
3. Richard and John have left leg. Left leg (Richard, John)
4. All students have a roll no. $\forall x$: students (x) → Roll no (x)
5. None of the students like Icecream. $\sim \exists x$: (students (x) → likes (x, icecream)
6. Every student likes healthy food. $\forall x$ (student (x) → likes (x, healthy fo
7. some students like ice cream. $\exists x$: (student (x) ∧ like (x, icecream)

(Pg 82) (to 134) Laws of Algebra & De-morgans law

Good Write

Universal instantiation :- $\forall x\, P(x) \therefore P(A)$ — A instance of

Existential Generalization :- $-P(A) \therefore \exists x P(x)$

$-P(A) \wedge P(B) \cdots \therefore \forall x P(x)$ [Universal generalisation]

[Existential instantiation] $\exists x P(x) \therefore P(F)$

DATE: __/__/____

PAGE: _____

$\forall x\, Q(x)$ / $Q(a)$

# Unification with example → Similar sentences that makes them looks the same and computes the substitution

UNIFY (Knows (John, x), Knows (John, Jane)) = {x / Jane}

# Generalized Modus Ponens (GMP)

$\frac{A \Rightarrow B, A}{B}$

$A_1'$ is King (John)

only John is King

$A_1 \wedge A_2 \wedge A_3 \cdots A_n \Rightarrow B, A_1', A_2', \cdots A_n'$

Sub (σ, B)

# Forward chaining

~~Subst~~

SUBST (σ, $A_i$) = SUBST (σ, $A_i'$)

# Inferences unification resolution

Slide 40 / 1st part

CNF

Algo ⇒ Putting Axioms into clausal form

1. biconditional / Implication :- $\rightarrow$ / $\Rightarrow$ / $\Leftrightarrow$ replace with $\vee$ + $\neg$
2. Move $\neg$ inward. Ex- $\neg\forall$ ($\neg$Animal(y) $\vee$ Loves(x, y)) ⇒ $\neg\neg$ Animal(y) Loves(x, y)

# Constraint Satisfaction

# Cryptarithmetic

Wednesday
25/10/2023

Good Write