UNIT-II

## Procedural knowledge

Procedural knowledge also known as Interpretive knowledge, Is the Type of knowledge In which It clarifies how a particular thing can be accomplished. It is not so popular because It is generally not used.

It emphasize how to do something to solve a given problem

example;

```
var a = [1, 2, 3, 4, 5, 6];
var b = [];
for (var i=0; i<a.length; i++)
{
    b.push(a[i]);
}
console.log(b);
```

Out put is

[1, 2, 3, 4, 5]

## Declarative knowledge

Declarative knowledge also known as Descriptive knowledge, Is the Type of

Knowledge which Tells the Basic knowledge about something and It is more than procedural knowledge.

example:

```
var a = [1, 2, 3, 4, 5];
var b = a.map(function(number)
{ return num * 1 });
console.log(b);
```

→ convert specified value into an Integer number

→ testing purpose

log()
method writes logs message to the console

Output:

[1, 2, 3, 4, 5].

→ Text Based
↓ entering the Text Input [computer KeyBoard]

In Both example we can See that the output of a given problem is same because the only difference in that two methods to achive the output or solution of problem.

Difference the procedural and Declarative knowledge.

| Procedural Knowledge | Declarative Knowledge |
|---|---|
| - It is also known as Interpretive knowledge. | It is also known as Descriptive knowledge. |

| Procedural Knowledge | Declarative knowledge |
| --- | --- |
| Procedural knowledge means how a particular thing can be accomplished. | Declarative knowledge means Basic knowledge about something |
| procedural knowledge is generally not used means It is not more popular | Declarative knowledge is more popular. |
| Procedural knowledge can't be easily communicate. | Declarative knowledge can be easily communicate. |
| Procedural knowledge is generally process Oriented in nature. | Declarative knowledge is Data oriented in nature. |
| In procedural knowledge debugging and validation is not easy | Declarative knowledge Debugging and validation is easy |
| Procedural knowledge is less effive in competitive Programming | Declarative knowledge is more effective in competitive programming |

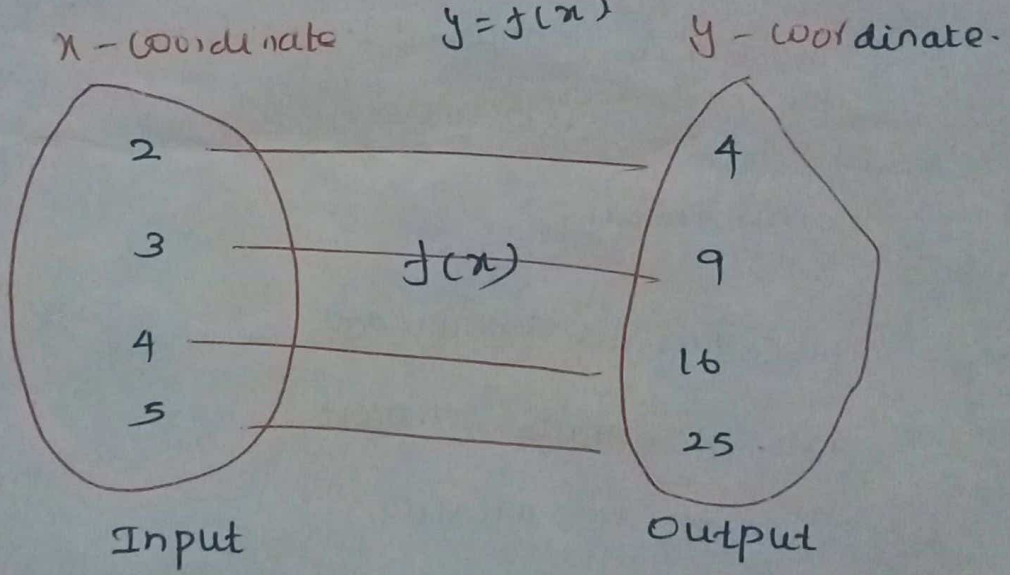
Console:

Entering the text Input through the computer keyBoard. By Reading the Text output from the computer Terminal.

var b= a.map( function (number)

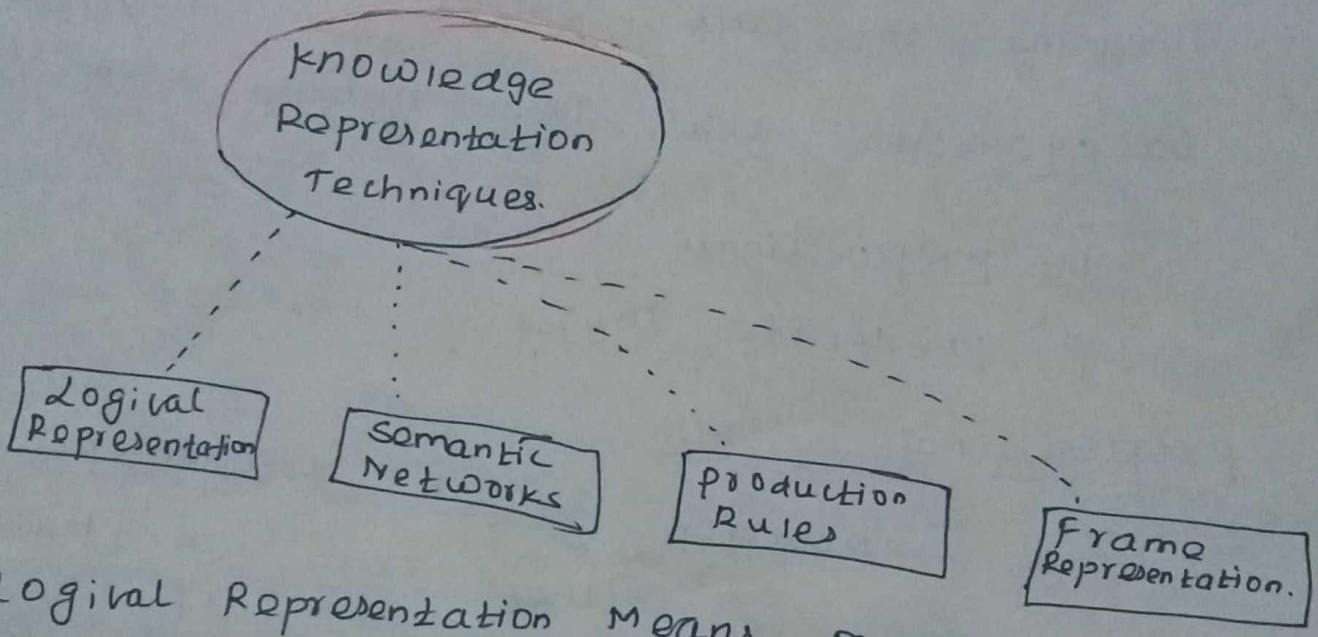
J/

Higher-order function that applies a given function to each element of collection. eg: list



## Knowledge Representation:

1. Logical Representation.
2. Semantic network Representation.
3. Frame Representation.
4. production Rules.




1. Logical Representation means Drawing a conclusion based on various conditions.

Syntax:

1. Syntaxes are the Rules which decide HOW we can construct legal sentences in the logic.
2. It determines which symbol we can use in knowledge representation.
3. How to write those symbols.

fx legal sentence:

- I have driven a car
- We wake up early in the morning.

Semantics:

- semantic also involves assiging a meaning to each sentence.

categorised into two mainly two logic:

1. propositional logics
2. predicate logics.

Propositional logics:

- proposition is a statement that can be either true (or) false.
- It must be one (or) the other, and It cannot be both.

Ex: $B = \{a_1, a_3\}$

M [given).

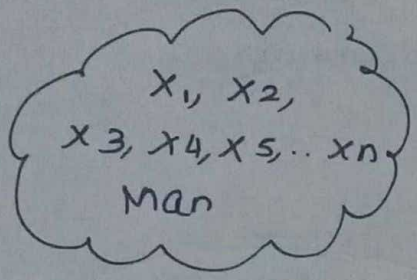
$M(a_1) = a_1$ $M(a_2) = a_2$

(and) $\wedge$, (or) $\vee$, (not) $\sim$, (Implies) $\Rightarrow$

If, x, y are propositions, so are:-

$x \wedge y$, $x \vee y$, $x \Rightarrow y$, $\sim x$

Predicate logic:




$x_1$ drinks coffee

$\wedge$

$x_2$ drinks

$\wedge$

$x_3$ drinks milk

$\vdots$

$\wedge$

$x_n$ drinks milk

we can write as

$\forall x$ man(x) -> drink (x, coffee).

$\vdots$

$\vee$

all x, where x is a man. who drinks coffee.

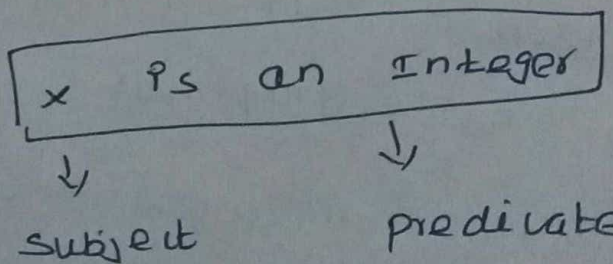
Example: 2:

All Birds fly

"fly( bird)"

$\forall x$ bird(x) -> fly(y)

predicate:



Advantages of logical Representation:-

1. Logical Representation enables us to do Logical Reasoning.
2. Logical Representation is the Basis for the programming language.

Dis Advantage:

1. Logical Representation Technique may not be very natural.
2. Logical Representation have some Restrictions.

Semantic Network Representation:

1. Semantic Networks are alternative of predicate logic for knowledge Representation.
2. In Semantic Networks, we can represent our knowledge in the form of graphical Networks.
3. this network consists of nodes representing objects and arcs which Describes the relationship between those objects.

4. Semantic networks are easy to understand and can be easily extend.

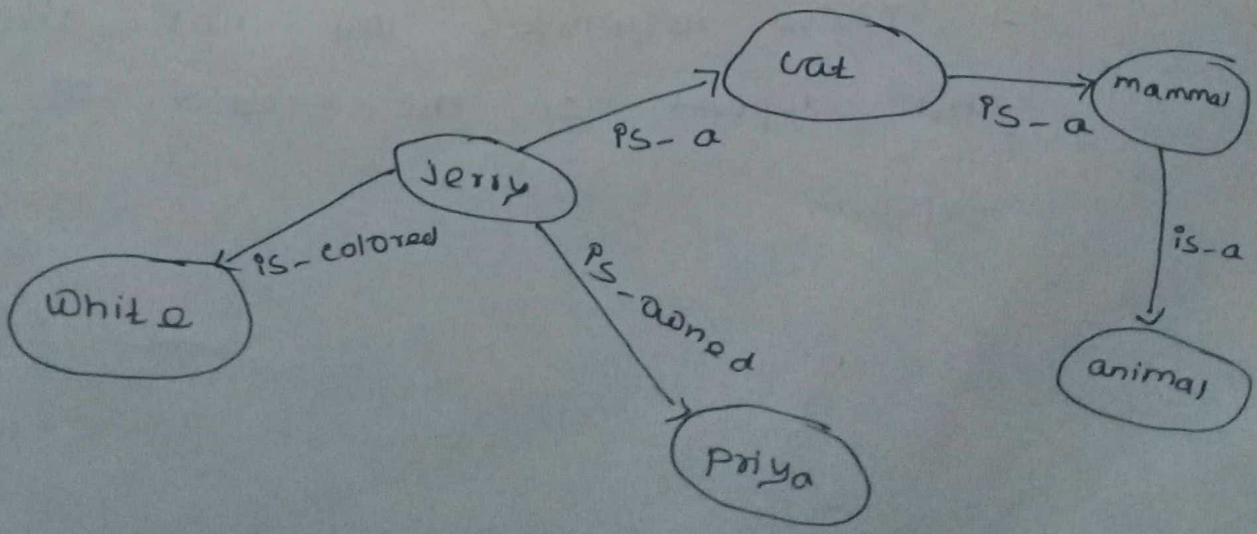
This Representation consist of mainly Two Types of relations:

a. Is-A relation (Inheritance)
b. kind of relation.

Example:

Following are some statements which we need to represent in the form of nodes and arcs.

a. Jerry is a cat
b. Jerry is a mammal
c. Jerry is owned by priya
d. Jerry is brown colored.



- In the above digram, we have represented the different type of knowledge in the form of nodes and arcs. Each object is connected with another object by some Relations.

Advantages:

1. These networks are simple and easily understandable.
2. semantic networks are a natural Representation knowledge.

Drawbacks:

1. Semantic networks do not have any standard definition for the link names.
2. These networks are not Intelligent and depend on the creator of the System.

Frame Representation:

1. It consists of a collection of slots and slot values.
2. These slots may be of any type and sizes.
3. The various aspects of a slot is known as Facets.
   ↓ Feature frame.
4. A Frame is also known as slot filler knowledge Representation in Artificial Intelligence.
5. A Single frame is not useful. Frames System consist of a collection of Frames which are connected.
6. In the frame, object (or) event can be

Example:

Let's take an example of a frame for a Book.

Slots — Filters

Title — Artificial Intelligence

Genre — Computer Science

Author — Peter Norvig

Eaition — Third Edition

Year — 1996

page — 1152.

## Advantages of frame Representation.

1. It is very easy to add slots for new attribute and relation.
2. It is easy to include default data and to search for missing values.

## DisAdvantages of frame Representation.

1. In frame System Inference mechanism is not be easily processed.
2. Frame Representation is much generalized approach.

4. Production Rules:

* Production rules system consist of [condition, action) pairs which mean- "If condition then action".
* It has mainly three parts
  1. The set of production rules
  2. Working memory
  3. The recognize_act-cycle.
* The production Rules Agents check condion
* The working memory contains the Description of the current state of problems-solving and Rule can knowledge to working memory.
* If there is a New situation (state) generates, Then multiple production rules will fired.

Example:

* IF (at bus stop AND bus arrives) action (get into the Bus)
* IF (on the Bus AND paid AND empty seat) The action (sit down)

- If (on bus AND unpaid) then Action (pay charges).
- If (bus Arrives at destination) then action (get down from the Bus).

Advantages of production rule:

1. The production Rules are expressed in Natural language.

Disadvantages of production rule:

1. Rule Based production systems are Inefficient.

# Forward chaining vs Backward chaining

## Forward chaining

Forward chaining starts from known facts and applies Inference rule to extract more data unil It reaches to the goal.

It is a Bottom-up approach

forward chaining is known as data driven Inference Technique as we reach to the goal using the available data.

## Backward chaining

Backward chaining starts from the Rule (goal) and works Backward through Inference rule to find the required facts that support the goal.

It is a Top-Down approach

Backward Chaining is known as goal-driven Technique as we start from the goal and devide into sub-goal to extract the facts.

| Forward chaining | Backward chaining |
| --- | --- |
| Forward chaining reasoning applies a Breadth-first search strategy | Backward chaining reasoning applies a Depth-first search strategy |
| Forward chaining Tests for all the available rules. | Backward chaining only Tests for Few required rules. |
| Forward chaining is suitable for the planning, monitoring, control and interpretation application. | Backward chaining is suitable for diagnostic, prescription and debugging application. |
| Forward chaining can generate an infinite number of possible conclusions. | Backward chaining generates a finite number of possible conclusions. |

Forward chaining

It operates in the forward direction

Forward chaining is aimed for any conclusion.

Backward chaining

It operates in the Backward direction.

Backward chaining is only aimed for the required data.

Example 1 [Forward chaining]

A

A → B

B.

He is running

If he is running, he sweats

He is sweating.

Example 2 [Backward chaining]

B

A → B

A

He is sweating

If he is running, He sweats

He is running.

Example 2:

As per the law, It is crime for an American to sell weapons to hostile nations. Vizar, an enemy of America has some missiles and all the missiles were sold to It By Robin, who is an american

facts:

" prove that robin is criminal"

① It is a crime for an American to sell weapons to hostile nations.

$$American(x) \wedge weapon(y) \wedge sells(x, y, z) \wedge hostile(z) \rightarrow criminal(x)$$

Note: Mention predicate logic.

② vizar has some missiles

$$owns(A, m) \wedge missile(m).$$

(A: vizar, m: missile)

predicate logic

③ All of the missiles were sold to country vizar by Robin.

(Previous)

missile(m) ∧ owns(A, m) → sells(Robin, m, A)

4) Enemy of America is known as hostile.

enemy(A, America) → hostile(A)

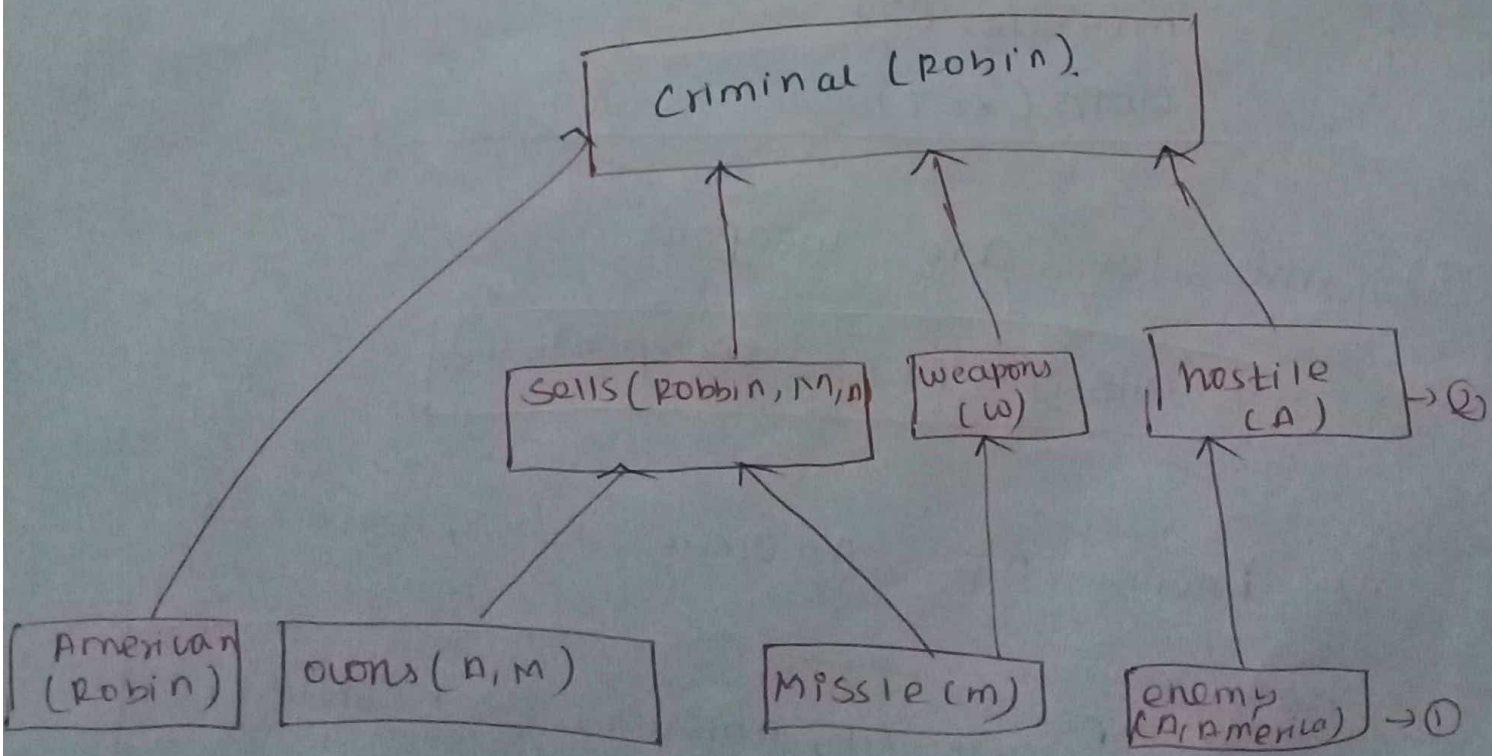
5) Missiles are weapons

missile(m) → weapons(w)

6) Country vizar is an enemy of America.

enemy(A, America)

Note: proof



# Backward Chaining

It is a crime for an American to sell weapons to hostile nations. Linda an enemy of America, has some missiles and all the missiles were sold to it by Robert, who is an American

prove; Robert is criminal

1) American(p) ∧ weapons(q) ∧ sells(p,q,r) ∧ hostile(r) → criminal(p).

2) Linda has some missiles

missiles(T)
owns(L, T).

3) missiles are weapons

missiles(T) → weapons(q)

4) Enemy of America is known as hostile.

enemy(E, America) → hostile(r)

5) Linda is an enemy of America.

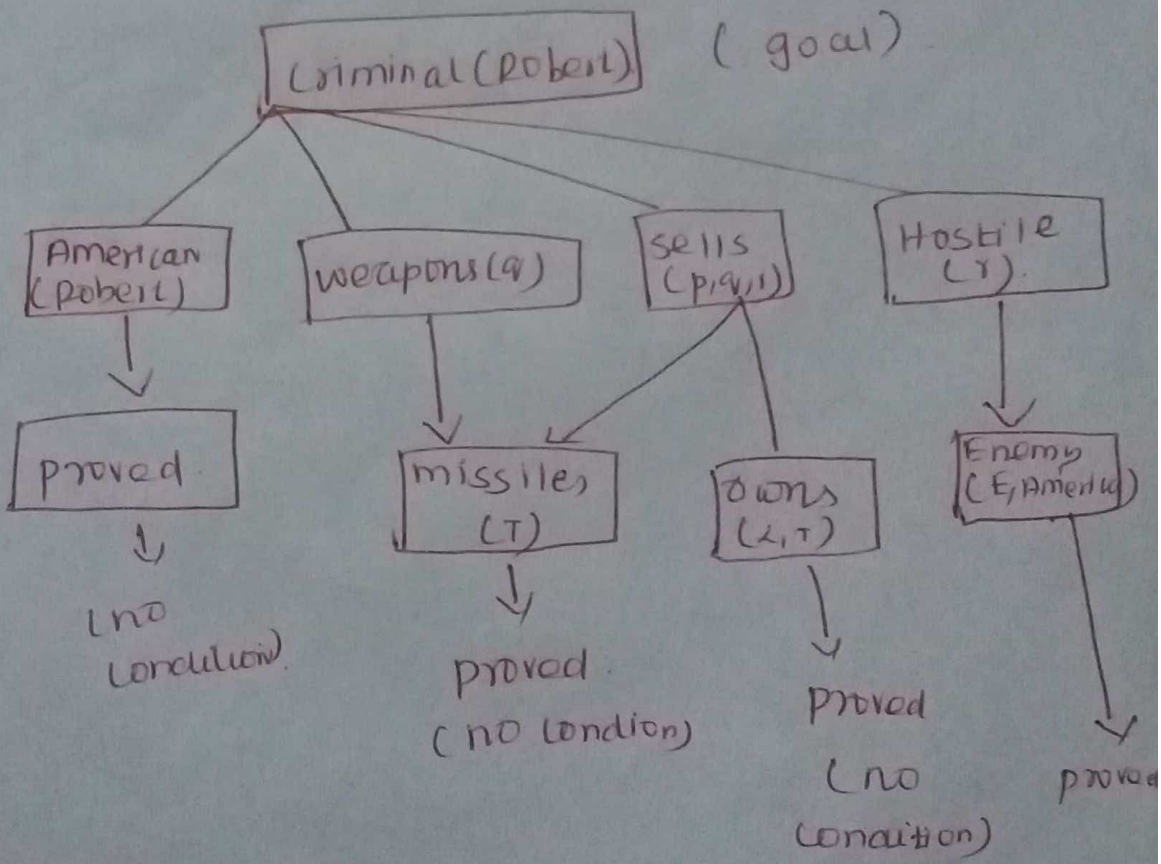
enemy (L, America)

6) Robert is an American.

American (Robert)

7) All the missiles were sold to country Linda by Robert.

missiles (T) ∧ Owns (L, T) → Sells (Robert, T, L)

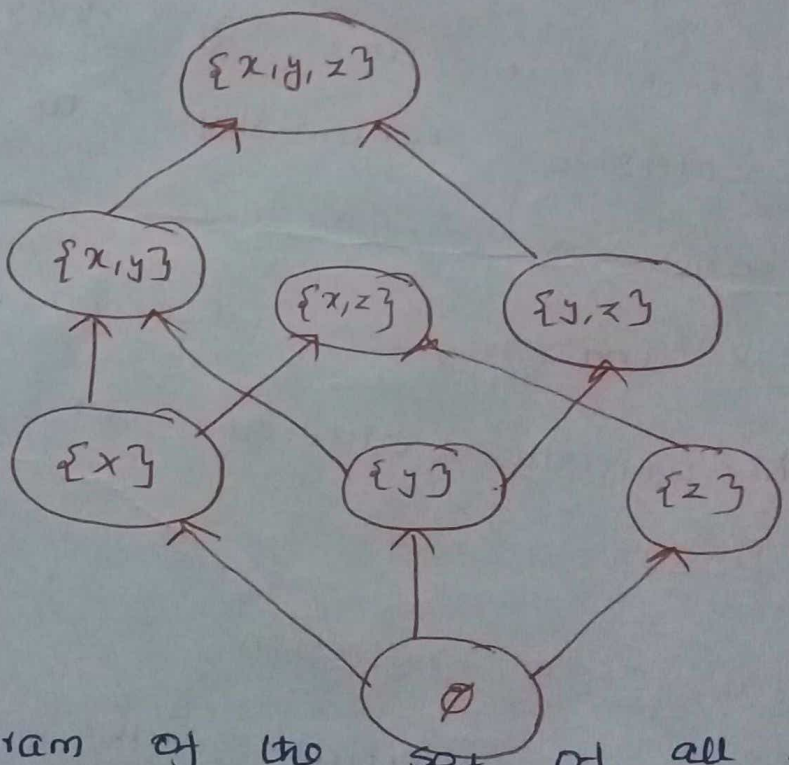



→ Represent fact

## Partial matching

Partial order planning is an approach to automated planning that maintains a partial order between actions and only commits ordering between actions, when forced to that is, ordering of action is partial.

- partial order generalize total orders. In which every pair is comparable.




- Hasse diagram of the set of all subset of three element set $\{x, y, z\}$. ordered by Inclusion. sets connected by an upward path like ∅ and $\{x, y\}$ are comparable, while eg; $\{x\}$ and $\{y\}$ are not.

* the word partial in the names "partial order" and "partially ordered set" is used as an indication that not every pair of elements needs to be comparable.

Informal definition:-

- partial order defines a notation of a comparison.

- Two elements x and y may stand in and of four mutually exclusive relationships to each other either $x < y$ or $x = y$, or $x > y$ (or x and y are incomparable.

* partially ordered set (also called a poset).

Partial order relation:

* A partial order relation is homogenous relation that is transitive and antisymmetric.

Ex

partial order Relation:-

- We consider a poset of a 3 Tuple $(P, \leq, <)$ where $\leq$ is a non-strid partial order relation.

proof:- $a, b, c, \in P$ It must satisfy

1) Reflexivity:- $a \leq a$ (ie, every element is related to itself.

2) Antisymmetry:- if $a \leq b$, and $b \leq a$ then $a = b$. (no two distinct elements precede each other).

3) Transitivity:- if $a \leq b$ and $b \leq c$ and $a \leq c$.

- non strit partial order is also known an antisymmetric preorder

Strit partial order:-

strid partical order is homogenous relation $<$ on a set P.

- Irreflexivity :- Not $a < a$ (ie no element is related to itself (also called anti-reflexive)

2. Asymmetry :- If $a < b$ then not $b < a$.

3 Transitivity :- If $a < b$ and $b < c$ and $a < c$.

note :- * Irreflexivity and Transitivity together imply Asymmetry.

* Asymmetry Implies Irreflexivity

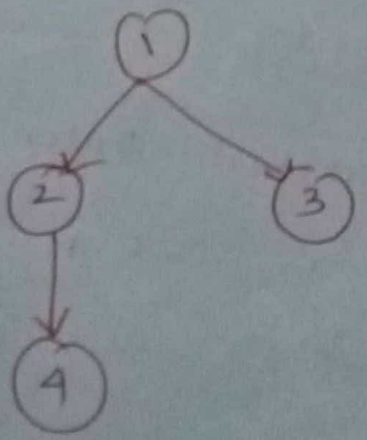
notation :-

$(P, \leq, <)$

Relation, $\leq$, $<$, $\geq$ and $>$

write $(P, \leq)$ or $(P, <)$

Example,



- Graph of the divisibility of numbers from 1 to 4.
- The set is partially but not totally ordered.
- Because there is relationship from 1 to every number.
- But not relationship to 2, 3, and 4.

Partial order planning; Example: 2

putting on a pair of shoes.

Goal:

Goal (RightShoeOn ∧ LeftShoeOn)

Init()

Action (RightShoe, PRECOND: RightSockOn, EFFECT: RightShoeOn)

Action (RightSock, EFFECT: RightSockOn)

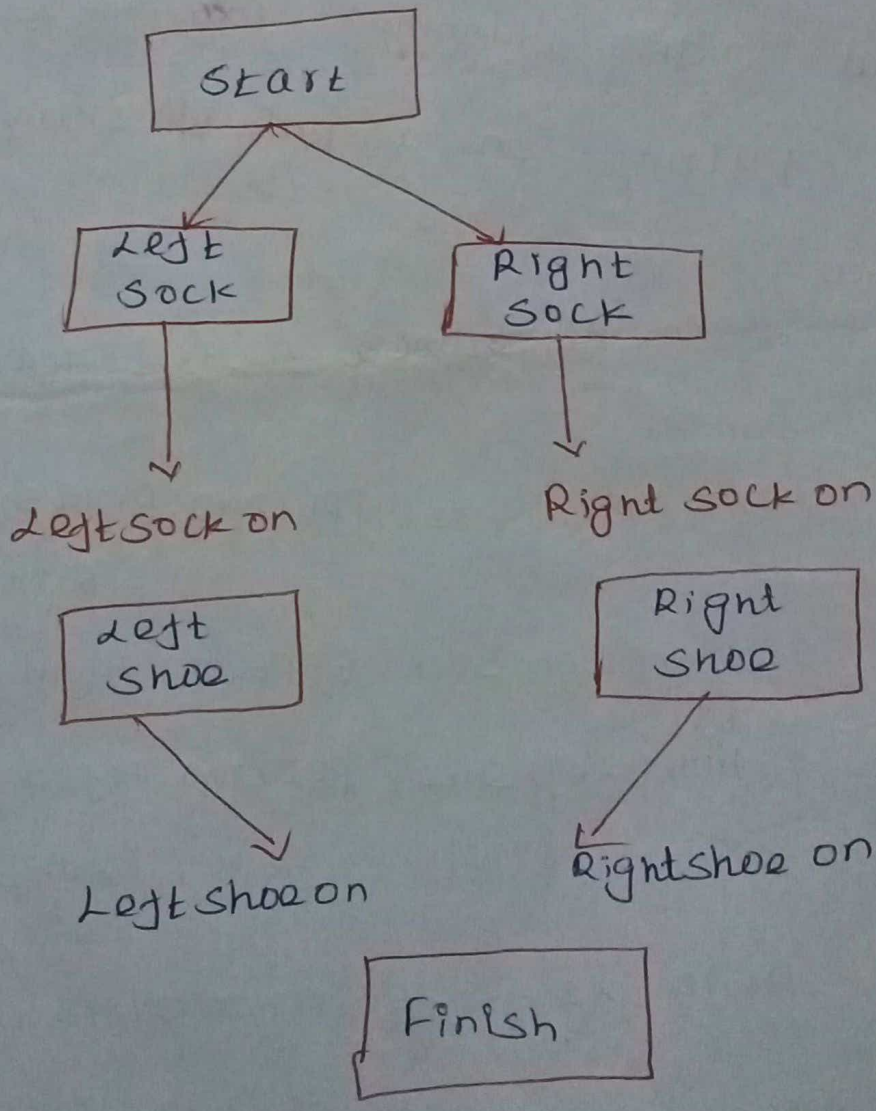
Action (LeftShoe, PRECOND: LeftSockOn, EFFECT: LeftShoeOn)

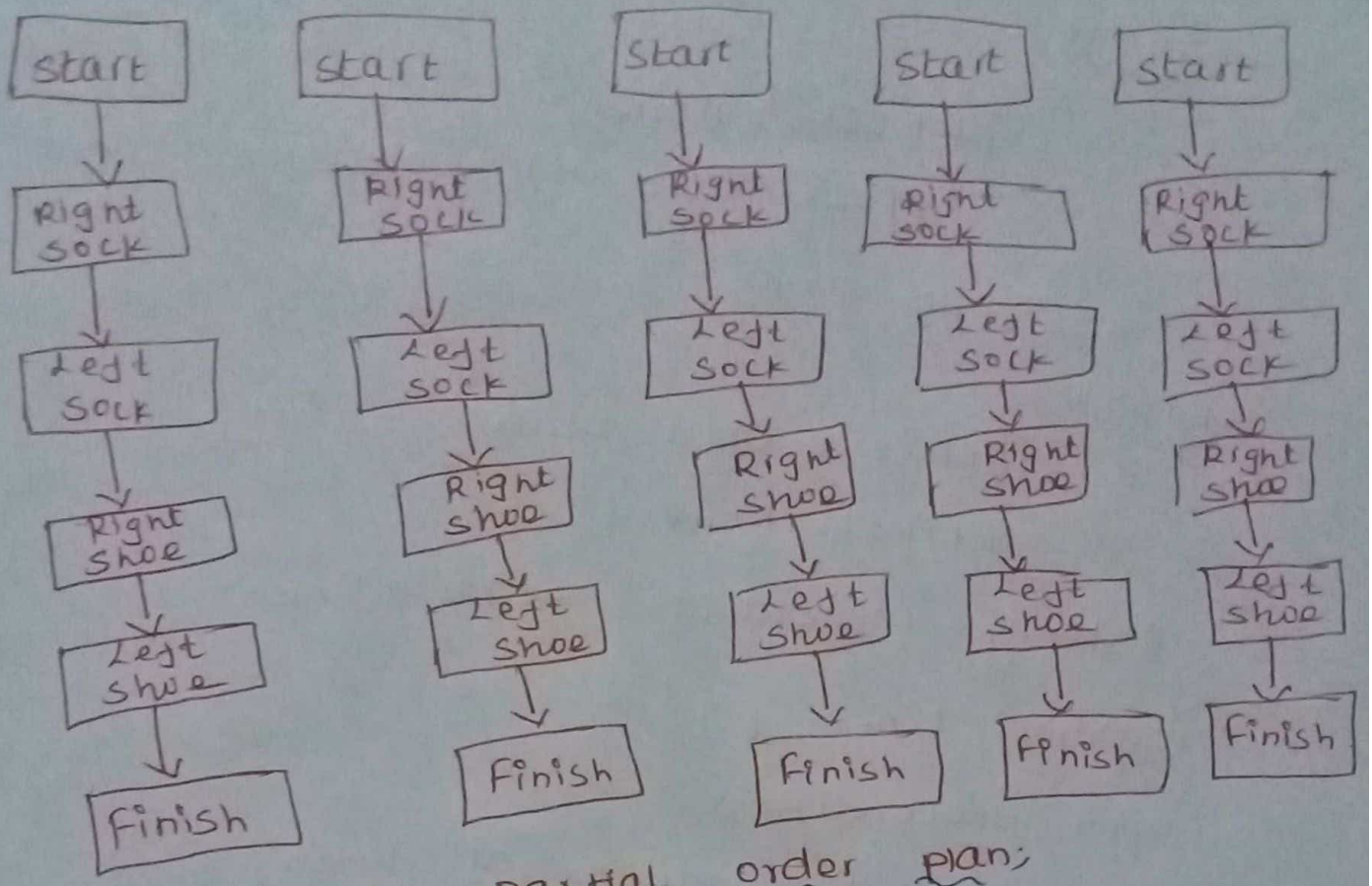
Action (LeftSock, EFFECT: LeftSockOn).

* A planner should be able to come up with two action sequence.

* RightSock followed by RightShoe to achiro the first conjunct of the goal
* LeftSock followed by LeftShoe for the second conjunct
* Then the two seaquences can be combined to yield to final plan.

Partial order Plan:-






How to define partial order plan;

- A set of actions, that make up the Make sets of the plan
- A set of ordering constrain

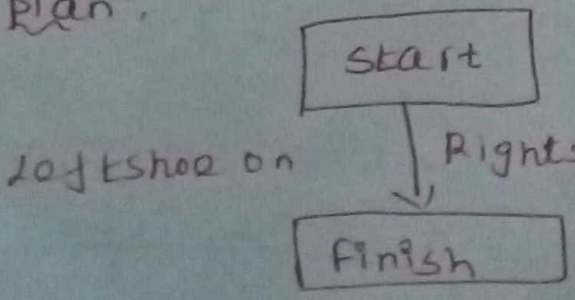
$A \prec B$

- A set of casual links

$A \xrightarrow{P} B$

Right Sock $\xrightarrow{\text{RightSockon}}$ Rightshoe

The Initial Plan:



- Start:
  - PRECOND; none
  - EFFECT; Add All propositions That are Initially True
- Finish:
  PRECOND; Goal State.
  EFFECT; None
- ordering constraints; Start < finish
- casual links; { }
- open preconditions:-
  - {preconditions of finish}

Example:

Final plan

-The final plan has the following the components

- Action:- { Right Sock, RightShoe, LeftSock, LeftShoe, Start, finish }
- Orderings; { RightSock < RightShoe, LeftSock < LeftShoe }
- open preconditions: { }
- Links:-

RightSock $\xrightarrow{\text{RightSockOn}}$ RightShoe

LeftSock $\xrightarrow{\text{LeftSockOn}}$ LeftShoe

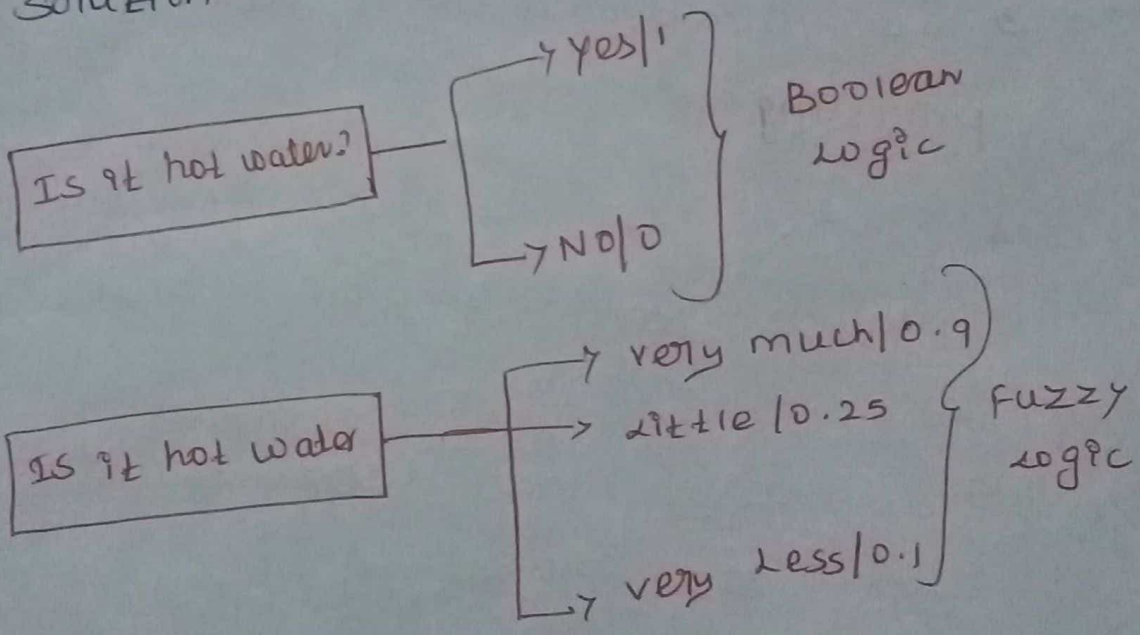
RightShoe $\xrightarrow{\text{RightShoeOn}}$ finish

LeftShoe $\xrightarrow{\text{LeftShoeOn}}$ finish

# FUZZY MATCHING ALGORITHMS

## Fuzzy Logic

- Fuzzy logic means the things that are not clear some times we cannot decide in real life that the given problem or statement is either true (or) false.
- At that time, this concept provides may values between the true and false and gives the flexibility to find the best solution to the problem.




* In the Boolean system, only two possibilities [0 and 1] exist, where 1 denotes the absolute truth value and 0 denotes the absolute false value.

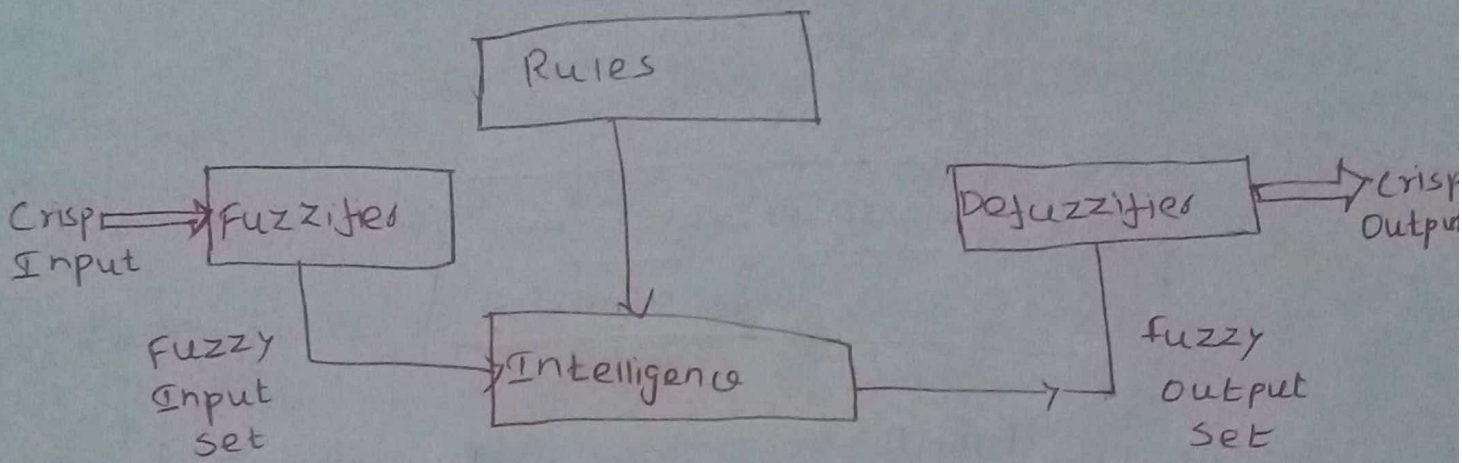
* But the Fuzzy System, there are multiple possibilities present between the 0 and 1 which are partially False and partially True.

## Characteristics of Fuzzy logic:

1. This concept is flexible and we can easily understand and implement it.
2. It is used for helping the minimization of the logics created by the human.
3. It is Based on natural language processing.
4. It is also used by the quantitative analysts for Improving their algorithm's execution.
5. It also allows users to Integrate with the programming.

## Architecture of a Fuzzy logic System

The architecture consists of the different four components which are given below.




Rules;

* It contains all the Rules and the if-Then conditions offered by the experts to control the decision making system.

Fuzzification:-

* This step converts Inputs or the crisp numbers into fuzzy sets.

* you can measure the crisp Inputs by sensors and pass them into the control system for further processing.

* It splits input signal into Five steps such as.

| | |
|---|---|
| LP | X is large positive |
| MP | X is Medium positive |
| S | Small |
| MN | Medium Negative |
| LN | X is large Negative |

Inference Engine:-

* It determines the degree of Match between fuzzy Input and the Rules

* According to the Input field, It will decide the rules that are to be fired.

* combining the fired rules from the control actions.

Defuzzification:-

* The Defuzzification process converts the fuzzy sets into a crisp value.

Example:

* The design of a Fuzzy logic System starts with a set of membership functions for each Input and a set for each output.

* A set of Rules is then applied to the membership functions to yield a crisp output value.

* Let's taken example of process control and understand Fuzzy logic.

Step 1;

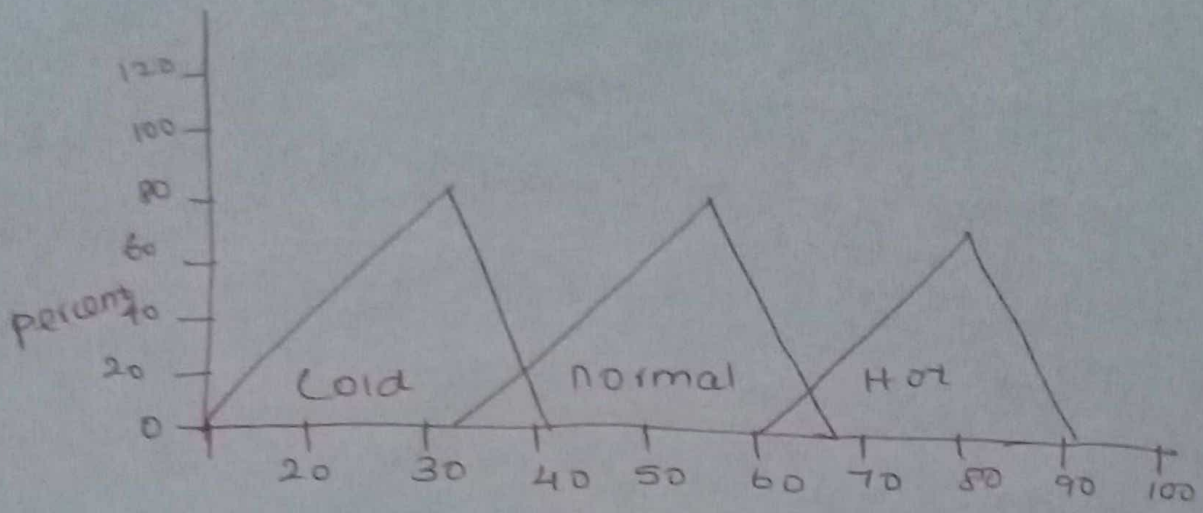
* Here, Tempreature is the Input and Fan Speed is the output. You have create membership functions for each Input.

* A membership function is simply a graphical representation of the fuzzy variable sets.

* for this example we will use three fuzzy sets, Cold, Warm, and Hot.

We will then create a membership function for each of three sets of

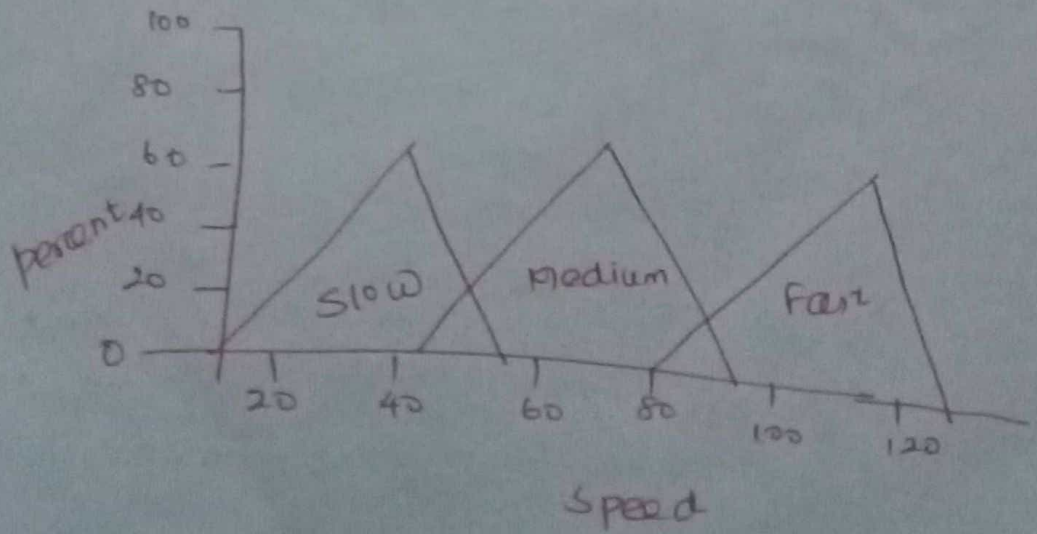
Tempreature.




Step 2:-

* In the next step, we will use three fuzzy sets for the output. Slow, Medium and fast.

* A set of functions is created for each output set just as for the Input sets.

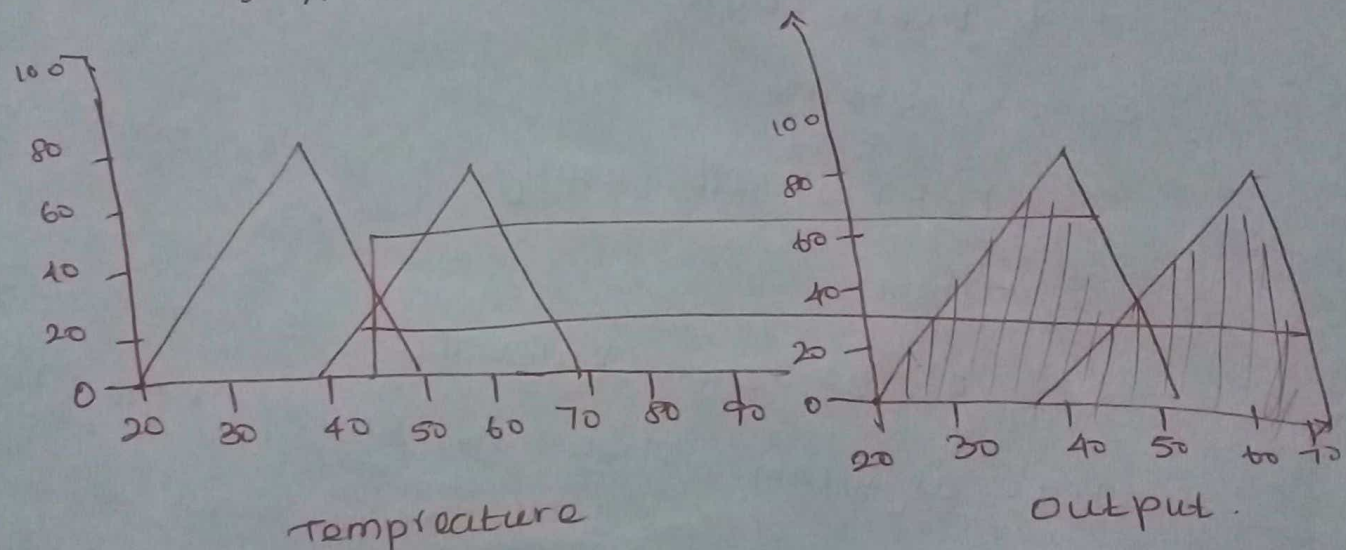


Step 3:

* We can create the Rules that will define how the membership functions will be applied to the final system

* we will create three Rules for this system

- If Hot then fast
- If Warm then Medium
- And, If Cold then Slow



## RETE algorithm:

* Rete algorithm is a pattern matching algorithm.

* It is a very efficient algorithm for matching facts against the patterns in rules.

## Rules, Ruleset and Facts:

* Ruleset is nothing But knowledge based consisting of one (or) more Business Rules (or simply Rules)

* Every Rule in the ruleset represents some knowledge.

* Rules are usally in the form of if-then.

* As if-then Rules are suitable for rete algorithm They are called rete rules.

* Here is a simple rete rule

```
If age > 60 then assign Status =
"Senior Citizen".
```

* where if, then and assign are keywords. if part represent condition and then part represents action.

* There may be more than one condition and in that case the conditions should be joined by logical operators

* clearly above example Rule means that if a person's age is more than 60 then he is a senior citizen.

* If we want to check whether a person is senior citizen or not we need to data that is the person age and this data is called fact

* Example:-

> if age > 60 and annual-Income < 12000
> then assign monthly-Bonus = 2000.

* To execute the above Rule we need two data or facts

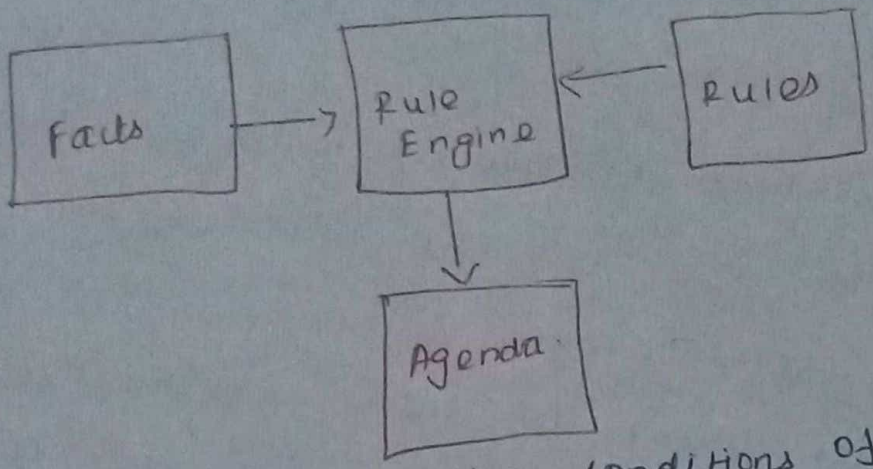
* one for age and one for annual Income.

* for checking the bonus of two persons we need two pair of data, and so on.

The Inference cycle:-

* Each Rule has an Inference cycle consisting of three parts.

* match, select, and execute.




* In matching phase, The conditions of the Rules are matched against the facts to determine which Rules are to be executed.
* The Rules whose conditions are met are stored in a list called agenda
* The action may be executing a user defined (or) built-in function (or) executing a decision Table (or) otherwise.

Note; That the decision Table engine is used to execute decision tables.

Building of the Rete Network;

* The Rete network is a Direct acyclic graph consists of nodes representing patterns in the condition of the Rules.
* The Rete network consist of Two parts
  * alpha network
  * beta network

* Alpha network consists of nodes known as alpha nodes. Each alpha node has one input that defines intra-element

* Beta network consists of beta nodes where each node take two Iput and to define Inter-element conditions

* The Rete network starts with the root node called Rete node.

* kind nodes follow the root node.

* There should be a kind node for each fact Type

* Alpha nodes are then created for each Pattern and connected to the corresponding kind node

* A condition, for example- a>b or b>c has two Patterns

* So two alpha nodes will be created one for a>b and one for b>c

* Each alpha node is associated with memory is called alpha memory.

* alpha nodes Then joined Beta nodes. Beta node accept only two Inputs

* So If there are three alpha nodes then first two nodes will be joined by one beta node.

* Thereafter, the output of this beta node and the third alpha node will be joined by another beta node.

* This way beta node supports partial matching.

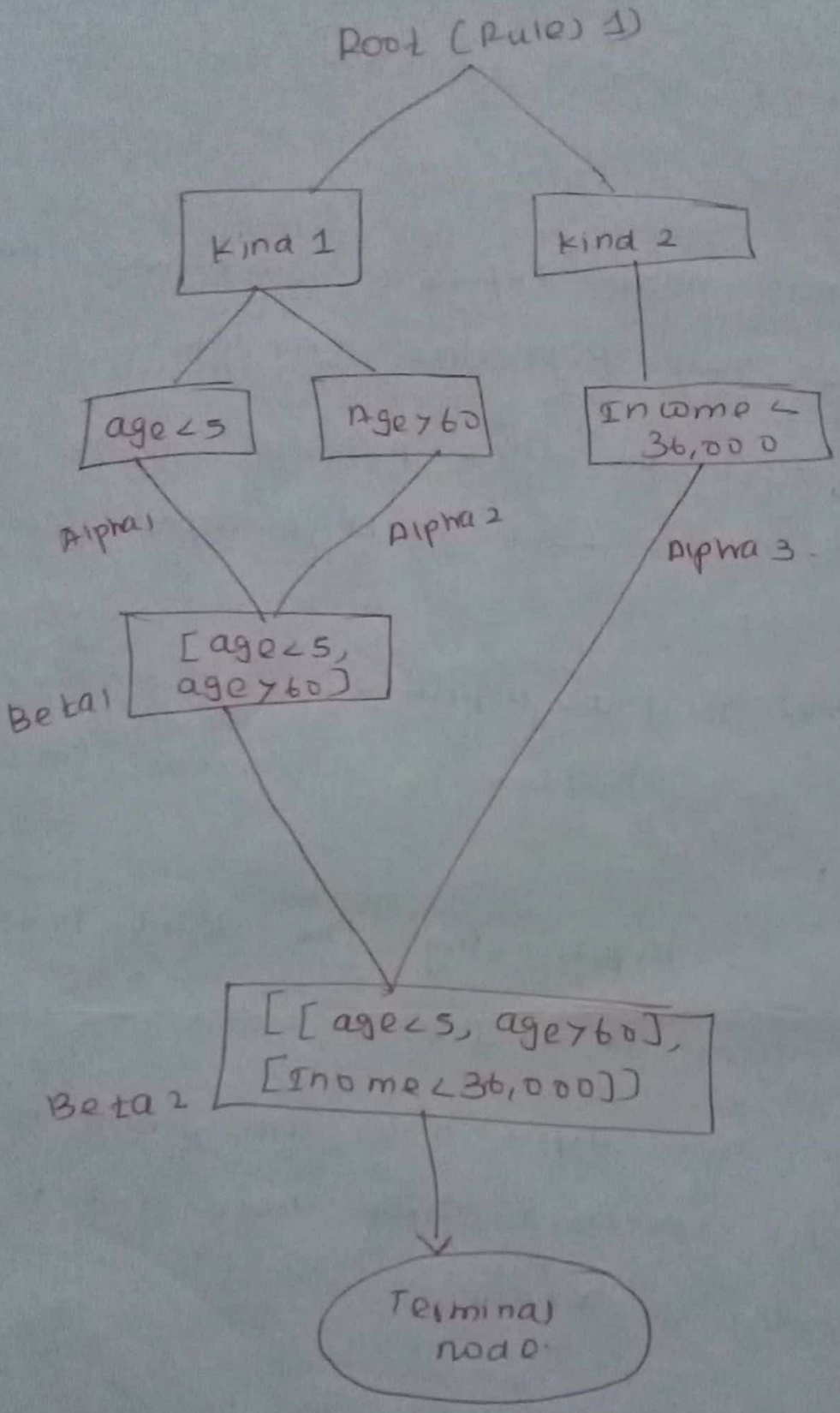
Example:-

Suppose we have a rule

If age > 60 (or) age < 5 (or) Income < 30,0000 then concession = 50/100

* Assume that age is java variable and Income is xml variable

* The following rete network can be created to represent this rule.


Root (Rule) 1)
Kind 1
Kind 2
age<5
Age>60
Income < 36,000
Alpha1
Alpha 2
Alpha 3
Beta1
[age<5, age>60]
Beta 2
[[age<5, age>60], [Income<36,000]]
Terminal node

* In the above rete network, there are two kind nodes as there are two types of facts.

* Kind 1 node Represents java type and kind 2 node Represents xml type.

* As there are three patterns; age<5, age>60 and Income <30000 three alpha nodes will be created.

* Alpha 1 and Alpha 2 Representing the two patterns is connected to kind 1.

* Alpha 3 Representing the third pattern is connected to kind 2

* First two alpha nodes are joined by beta 1. The third alpha node and beta 1 and joined by beta 2.

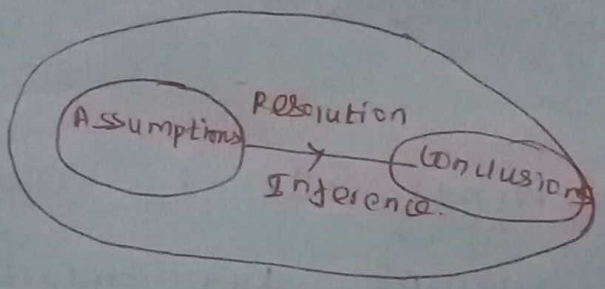
* value for age enters the root a token will be created.

* copies of token will be passed onto alpha 1 and alpha 2

## Logic programming:

* Logic programming is a method that computer scientists are using to try to allow machines to reason because it is useful for knowledge representation.

* In logic programming logic is used to represent knowledge and inference is used to manupulate it.




* The logic used to represent knowledge in logic programming is clausal form which is a subset of first-order predicate logic.

* It is used because first-order logic is well understood and able to represent all computational problems.

AI - programming language

* python
* R
* LISP
* Java
* C++
* Julia
* prolog

LISP

* LISP is one of the most efficient programming languages for solving specific problems.
* currently It is mainly used for machine learning and Inductive logic progiam problems.
* It has also influenced the creation of other programming language for AI and some examples are

* R
* Julia

Features of lISP:

* It is a machine- Independent language
* It allows us to create and update the programs and applications dynamically
* It provides high-level debugging
* It will support Input and output functions
* It is an expressed Based language.

Hello world program in lISP:

* we can start writing a string by using the write-line method.

Syntax:

(write-line string)

Example:

lisp

; this is a comment

write-line "Hello Geeks")

Naming conventions:-

* The naming conventions mean the way we are declaring variables in program
* It includes the variable names and syntax formats

Let us discuss the conventions:

* A variable can contain any number of alphanumeric characters other than whitespace, open and closing parentheses.

Example -

Acctable - hello, yuvan, etc

Not Acctable - hello(), sat {

sravab{ ,,, etc.

* A variable can not contain double and singie quotes, backlash, comma, colon, semicolon, and vertical bar.

Example -

Acctable_ hello, yuvan, etc

Not Acctable - hell " ") 0, sat // * &

sra//yvab{ ... etc.

* A variable can not start with a digit but it can contain any number of digits.

Note:- TERPRI function (the name stands for "Terminate printing", as it's Intended to be used to Terminate a line

of output

Example:-

Acceptable- hello88geeks, r56LISP, ,,,, etc

Not Acceptable - 40 geeks, 4kill, ..etc

Example code: For acceptable names

```
; acceptable naming conventions
(write-line "hello")
(terpri)
; acceptable naming conventions
(write-line "hello99")
terpri
" acceptable naming conventions
(write-line "hello geeks")
(terpri)
" acceptable naming conventions
(write-line "hello-geek")
(terpri)
" acceptable naming conventions
(write-line "hello123")
```

output:-

Hello

Hello99

Hello geeks

Hello_geek

hello123.