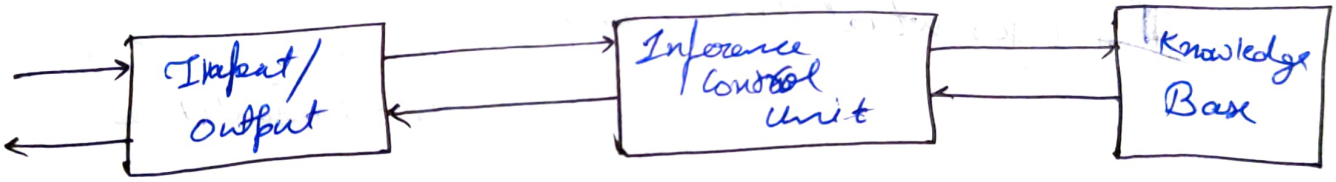
12/10/22

Knowledge Based System: These are the system that depends on a rich base of knowledge to perform difficult tasks. It includes work in vision, learning, general problem solving, & Natural language Understanding.

The systems get their power from the expert knowledge that has been coded into facts, heuristics & procedures. The knowledge is stored in the knowledge base that is separate from the control & inferencing components. It is possible to add new knowledge & refine existing knowledge without recompiling the control & inferencing programs.



Components of Knowledge Based System.

#Representation of Knowledge

① The objective of the knowledge Representation (KR) is to express knowledge in Computer tractable form. So that it can be used to enable our AI agents to perform as per the expectation

② A KR language is defined by 2 aspects.

- Syntax → The syntax of a language defines which configurations of the components of the language constitute valid sentences

- Semantics - The semantics defines what facts in the World the sentences refer to. Hence the statement about the world that each sentence makes.

Approaches to KR / Requirement of KR

① A good System for representation of knowledge in a particular domain should possess the following properties.

(I) Representational Adequacy → The ability to represent all the different kinds of knowledge that might be needed in that domain

(II) Inferential Adequacy → The ability to manipulate the representational str. in such a way as to derive new structures corresponding to new knowledge inferred from old ones.

(III) Inferential Efficiency → The ability to incorporate the additional info into the knowledge str. which can be used to focus the attention of the inference mechanism in the most promising directions

(IV) Acquisitional Efficiency → The ability to acquire new info. easily.

* Ideally the agents should be able to control its own KA. But direct insertion of info. by Knowledge Engineer would be acceptable.

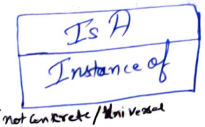
Techniques of KR

Object Oriented / Object Based
fully Obj oriented
5 ways to share windows

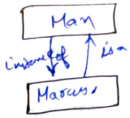
① Simple Relational Knowledge → It is used in database systems to represent declarative facts

Declarative Knowledge
passive knowledge expressed as statements of facts abt the world.

② Inheritable knowledge → It used the concept of property Inheritance.



* not concrete / universal



③ Inferential Knowledge → Uses the concept of Std. logical rules of inference.

Eg - Modus Ponens, Modus Tollens.

④ Procedural Knowledge → It is compiled knowledge related to the performance of some Tasks.

Eg → Algebraic Eqn Solving

→ Ordinary set Algebra ↔ Crisp set Algebra

Fuzzy Logic → {Unclear}

① Fuzzy System → They include fuzzy logic & fuzzy set theory.

(i) Knowledge exists in two distinct forms.
- → Objective Knowledge → that exist in Mathematical form & is used to represent engineering tools.
- → Subjective Knowledge → that exists in linguistic form & is usually imprecise to quantify.

(ii) The concept of fuzzy logic is used to coordinate these two forms of knowledge in a logical way.

(iii) Many real world problems have been Modelled, simulated & replicated with the help of fuzzy systems.

(iv) Some Applications of Fuzzy Systems are
- Info. Retrieval Systems
- Navigation Systems
- Robot Vision, etc

14/10/22

Defination of Fuzzy Logic

Fuzzy Logic is derived from Fuzzy Set Theory dealing with reasoning i.e approximate rather than precisely deduced from classical two valued logic.

Note → A declarative sentence that is either true or false is a proposition

Exception → Commands, questions & Exclamations.

Rules of Inference

① Modus Ponens → $(p \wedge (p \to q)) \to q$

② Modus Tollens → $(\neg q \wedge (p \to q)) \to \neg p$

In fuzzy logic, the truth values are multi-valued such as absolute True, Partially true, Absolute False, etc.

Fuzzy Proposition → A fuzzy proposition is a statement, $\tilde{P}$, which aquires a fuzzy truth value $T(\tilde{P})$

Fuzzy Connectives →

① Negation : $1 - T(\tilde{P})$

② Disjunction : $\max[T(\tilde{P}), T(\tilde{Q})]$

③ Conjunction : $\min[T(\tilde{P}), T(\tilde{Q})]$

④ Implication / Conditional : $\tilde{P} \Rightarrow \tilde{Q} = \neg P \vee Q$
: $\max[1 - T(\tilde{P}), T(\tilde{Q})]$

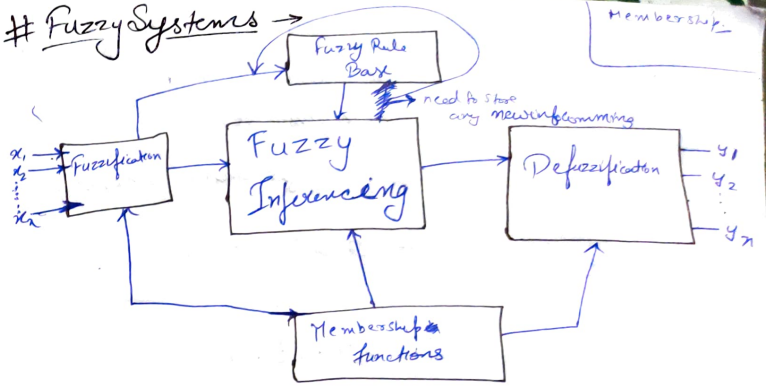
Eg → $\tilde{P}$: Mary is efficient
$\tilde{Q}$: Ram is efficient
Given → $T(\tilde{P}) = 0.8$
$T(\tilde{Q}) = 0.65$

Cal (i) $\neg\tilde{P} = 1 - 0.8 = 0.2$
(ii) $\neg\tilde{Q} = 1 - 0.65 = 0.35$
(iii) $\tilde{P} \wedge \tilde{Q} = \min[0.8, 0.65] = 0.65$
(iv) $\tilde{P} \vee \tilde{Q} = \max[0.8, 0.65] = 0.8$
(v) $\tilde{P} \rightarrow \tilde{Q} = \max[0.2, 0.65] = 0.65$

Fuzzy Quantifiers → In crisp logic, the predicates are quantified by quantifiers. Similarly, In fuzzy logic the propositions are quantified by quantifiers. There are two types

(i) Absolute Quantifiers (around about)
(ii) Relative Quantifiers (almost)

Fuzzy Systems →



Block Schematic Diagram

Fuzzy System Elements →

② Input Vector → These are the crisp values which are transformed into Fuzzy Sets in the fuzzification block.

$$X = [x_1, x_2 \ldots x_n]^T$$

② Output Vector → It comes out from the defuzzification block which transforms an output fuzzy set back to crisp value.

$$Y = [y_1, y_2 y_3 \ldots y_n]^T$$

③ Fuzzification → It is a process of transforming crisp value into grades of Membership for linguistic terms.

Such as → far, small, near of fuzzy sets.

④ Fuzzy Rule Base → It is a collection of Proposition containing linguistic variables.

IF → THEN

Eg → IF (x is A) (y is B) THEN (z is C)

variables

Where A, B, C → linguistic Variable / Terms

⑤ Membership Function → It provide the measure of the degree of similarity of elements in the universe of Discourse (U). to fuzzy set.

⑥ Fuzzy Inferencing → It combines the facts obtained from the fuzzification with the Rule Base & conducts the fuzzy reasoning process.

⑦ Defuzzyfication → It translates the results back to the real world values.

In many situations, for systems whose o/p is fuzzy - it is easier to take a crisp decision if the o/p is represented as a single quantity.

The typical defuzzification Methods are

① Centroid Method
② Centre of Sums
③ Mean of Maxima

17/10/22

First Order Logic

for Unit-2 → Richen Knight

Self Read → Write Short notes on Knowledge Based Agents & Read about the example of Wumpus World [Russel - Norvig]

First order Logic / First Order Predicate Logic

- First order logic is another way of KR in AI
It is an extension to propositional logic. It is a powerful language that develop info. abt the objects in a more easy way & can also express the relationship b/w those objects [and also functions]

Syntax of FOL

Some basic elements of FOL are.

1. Constants
2. Variables
3. Predicates → (Brother, Sisters, Fathers)
4. Functions → (sqrt, cuberoot)
5. Connectives
6. Equality (=)
7. Quantifiers → (forall, there exist)

Atomic Sentences → most basic sentences of FOL & are formed from a predicate symbol followed by parenthesis with a sequence of terms & represented as.

Predicate (term1, term2, . . . , term n).

Forex → Hari & Raghu are Brothers

Brothers (Hari, Raghu)

(2) Tommy is a Dog

Dog (Tommy)

Complex Sentences → It is made by combining atomic sentences using connectives

FOL Statements are divided into two parts

(1) Subject → main part of statement.

(2) Predicate → Relationship which binds the atoms together in a statement.

For Eg → X is a Integers

Subject → Predicate

Quantifiers → It is a language which generates quantifications

They are the symbols that permit to determine the identity of variables in terms of range & scope in the logic expressions

Quantifiers

- Universal (range) [for all, for every, for each]
- Existential (scope) [there exist, for some]

Universal Quantifiers → ∀ is the symbol of logical Representation which specifies that the statement within its range is true for everything & every instance of a particular thing.

∀ is represented as (∀) * uses implication

Existential Quantifiers → It is the type of quantifier which expresses that the statement within its scope is true for atleast one instance of something.

Represented as (∃) * uses And (∧)

Eg → ① All men drink coffee

x → Variable

↳ x1 coffee ∧ x2 coffee ∧ ... ..

↳ ∀ x man(x) ⟹ drink (x, coffee)

atomic

* All birds fly — ∀x bird(x) → fly(x)

* Every Man ~~respects~~ respects his parents → respects (x, y)

Man Parent

∀x Man(x) ⟹ respects (x, Parents)

∀y man(x) ⟹ respects (x, y)

Eg → Some boys are intelligent.

x → Variable

x1 intelligent ∨ x2 int.

∃ x : boy (x) ∧ intelligent (x)

Horn clause

Inference in FoL → It is used to deduce new facts & sentences from existing ones

Terminologies → ① Substitution → It is a fundamental operation performed on terms & formulas. It occurs in all inference systems in FoL.

F [a/x] → Subs. a for x

↓

replace x by a.

② Equality → Brother of John = Smith

↳ Brother (John) = Smith

Eg → ∩ x = y ⟹ x ≠ y.

Inference Rules for Quantifiers →

① Universal Generalisation → It is a valid inference rule which states that if P(c), premise, is true for any arbitrary element c in the universe of discourse then ∀ x P(x) is the conclusion.

② Universal Instanciation / Universal Elimination

It can be applied multiple times to add new sentences, we can infer any sentence P(c) by substituting a ground term c (must pick c from a Domain) from ∀x P(x) for any object in the universe of Discourse

$$\frac{\forall x\, P(x)}{\therefore P(c)}$$

Eg → Every person likes Ice cream

③ Existential Instanciation / Existential Elimination

It is a valid inference Rule in FOL & can be only applied once to replace the existential sentence. One can infer P(c) from the formulae in the form:

$$\frac{\exists x\, P(x)}{\therefore P(c)} \quad \text{(for a new const. } c\text{)}$$

④ Existential Generalization / Existential Introduction →

If there is some element c in the universe of discourse which has a property P then we can infer that there exist something in the universe which has prop. P.

$$\frac{P(c)}{\therefore \exists x\, P(x)}$$

Unification → It is all about making the expressions look identical.

L_1: P(x, F(g))
L_2: P(y, F(z))
→ Identical if x = y & g = z
(y/x, z/g)

Conditions for Unification → ① Predicate Symbol must be same, Atoms or expression with diff predicate symbols can never be unified.

② no. of args in both exp must be identical.

③ Unification will fail if there are two similar variables present in the same expression.

Unification Algo →

Unify (L_1, L_2)

1. if L_1 or L_2 is a variable or constant then
   (a) if L_1 & L_2 are identical → return [NIL]
   (b) else if L_1 → variable then
      i) if L_1 occurs in L_2 then return [FAIL]
      else return (L_2/L_1)
   (c) else if L_2 → variable then
      i) if L_2 occurs in L_1, then return {FAIL}
      else return (L_1/L_2)
   (d) else return {FAIL}

2. If initial predicate symbol in L_1 & L_2 are not identical then return {FAIL}.

3. If L_1 & L_2 have diff no. of args then return {FAIL}

4. Set SUBST to NIL

5. Loop for i=1 to no of args in L_1
   (a) Call Unify with i^{th} arg of L_1 & i^{th} arg of L_2. in S. [S = Unify(Z_i, Y_i)]
   (b) If S = FAIL. then return {FAIL}
   (c) If S ≠ NULL then
      (i) Apply S to the remainder of both L_1 & L_2
      (ii) SUBST = APPEND(S, SUBST)

6. return SUBST.

Implementation of Unification Algo →

1. Initialise the SUBST set to be empty.
2. Recursively unify atomic sentences
   (a) Check for identical exp. match
   (b) If one exp is a variable V_i & other is a term t_i which does not contain the variable V_i
      then (i) Subs (t_i/V_i) in existing substitutions
      (ii) Add t_i/V_i to the substitution set list.
      (iii) If both the exp are functions then function name must be similar & no of args must be same in both exp.

~~P(Z,A)~~ P(x, g(x))

① P(Z, y) → Unification possible Z/x, y/g(x)

② P(Z, g(Z)) → Z/x, g(Z)/g(x)

③ P(prime / ± (prime)) → Not

Resolution is a single inference rule, which can efficiently operate on CNF or clausal form.

CNF (Conjunctive Normal Form)

CNF → It is a sentence represented as a conjunction of clauses.

Clausal form → Disjunction of literals is called clause

Steps for resolution.

1. Conversion of facts into first order logic
2. Convert first order logic statements into CNF.
3. Negate the statement, which needs to be proven (by contradiction)
4. Draw a resolution graph (Unification)

Eg →
1. If it is Sunny and warm day, you will enjoy.
2. If it is raining, you will get wet.
3. It is a warm day.
4. It is raining.
5. It is Sunny.

To Prove → You will enjoy.

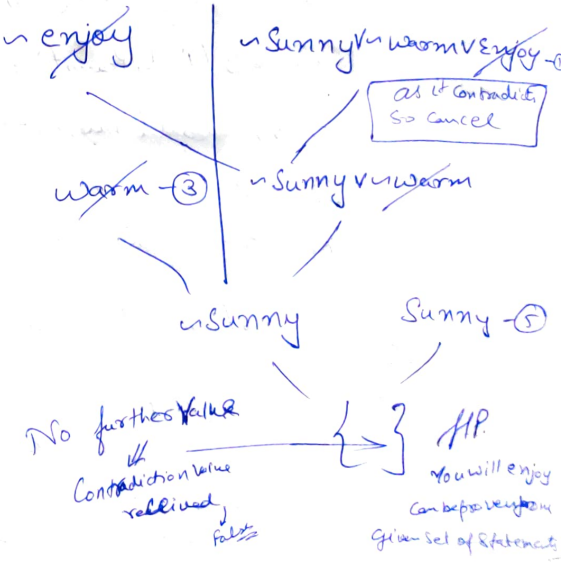
First order vs Second Order

Step-① →
1) Sunny ∧ Warm ⇒ Enjoy.
2) Raining ⇒ wet
3) Warm
4) Raining
5) Sunny

Step-② →
1) ~(Sunny ∧ warm) ∨ Enjoy → (Demorgan) → ~Sunny ∨ ~Warm ∨ Enjoy.
2) ~raining ∨ wet
3) warm
4) Raining
5) Sunny.

Step-③ → ~Enjoy

Step-④ → ~enjoy



Eg → St① → John likes all kind of food

St② → Apple & Vegetable are food.

St③ → Anything anyone eat & not killed is food.

St④ → Anil eat peanuts & is still alive.

St⑤ → Harry eats everything that Anil eats.

St⑥ → John likes peanuts. [To Prove]

St⑥ → If not killed then alive

St⑦ → If alive then not killed.

Step ① →

1) $\forall x$: food(x) → likes(John, x)
2) food(Apple) ∧ food(Vegetable)
3) $\forall x \forall y$: eats(x, y) ∧ ~killed(x) → food(y)
4) eats(Anil, Peanuts) ∧ alive(Anil)
5) $\forall x$: eats(Anil, x) → eats(Harry, x)
6) $\forall x$: ~killed(x) → alive(x)
7) $\forall x$: alive(x) → ~killed(x)

⑧ → likes(John, Peanuts)

*CNF → (a) Eliminate all implications & rewrite

(b) Move negations inwards & rewrite.

(c) Rename variables or Standardise the variables

(d) Eliminate Existential Quantifiers. [These exist ko hatao]
Instances

Step ② → (a)

1) $\forall x$: ~food(x) ∨ likes(John, x)
2) food(Apple) ∧ food(vegetable)
3) $\forall x \forall y$: ~(eats(x, y) ∧ ~killed(x)) ∨ food(y)
4) eats(Anil, Peanuts) ∧ alive(Anil)
5) $\forall x$: ~eats(Anil, x) ∨ eats(Harry, x)
6) $\forall x$: ~(~killed(x)) ∨ alive(x)
7) $\forall x$: ~alive(x) ∨ ~killed(x)

(b) De morgan

1) $\forall x$: ~food(x) ∨ likes(John, x)
2) food(Apple) ∧ food(vegetable)
3) $\forall x \forall y$: ~eats(x, y) ∨ killed(x) ∨ food(y)
4) eats(Anil, Peanuts) ∧ alive(Anil)
5) $\forall x$: ~eats(Anil, x) ∨ eats(Harry, x)
6) $\forall x$: killed(x) ∨ alive(x)
7) $\forall x$: ~alive(x) ∨ ~killed(x)

(c) →

1) $\forall x$: ~food(x) ∨ likes(John, x)
2) food(Apple) ∧ food(Vegetable)
3) $\forall y \forall z$: ~eats(y, z) ∨ killed(y) ∨ food(z)
4) eats(Anil, Peanuts) ∧ alive(Anil)
5) $\forall w$: ~eats(Anil, w) ∨ eats(Harry, w)
6) $\forall p$: killed(p) ∨ alive(p)
7) $\forall q$: ~alive(q) ∨ ~killed(q)

(e) Drop Universal Quantifiers

(d)

1) $\neg food(x) \vee likes(John, x)$
2) $food(Apple) \wedge food(Vegetable)$
3) $\neg eats(y, z) \vee killed(y) \vee food(z)$
4) $eats(Anil, Peanuts) \wedge alive(Anil)$
5) $\neg eats(Anil, w) \vee eats(Harry, w)$
6) $killed(k) \vee alive(k)$
7) $\neg alive(q) \vee \neg killed(q)$

(f) Distribute Conjunction over disjunction

(1) $(A \wedge B) \vee (C \wedge D)$
(2) $(A \vee B) \wedge (C \vee D)$

[Make statements with $\wedge$ separate]

Step-2 → $\neg likes(John, peanuts)$

Step-1 → $\neg likes(John, peanuts)$ $\neg food(x) \vee likes(John, x)$ (1)

(peanuts/x)

$\neg eats(y, z) \vee killed(y) \vee food(z)$ (3) $\neg food(peanuts)$

peanuts/z

$\neg eats(y, Peanuts) \vee killed(y)$ $eats(Anil, Peanuts) \wedge alive(Anil)$ (4)

(Anil/y)

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

(Anil/k)

$alive(Anil)$ $\neg alive(Anil)$

{ } → NIL

21/10/22 Forward Chaining (Reasoning)

It is the form of reasoning which starts with atomic sentences in the knowledge base & applies inference rules in forward dir^n to abstract more data until the goal is reached.

Properties → ① It moves from bottom to Top

② It is the process of making a conclusion based on known facts or data by starting from initial state & reach the goal state.

③ Also k/a Data Driven Strategy as reaching to the goal uses available data

Applications → Expert Systems.

Backward Chaining (Reasoning)

It starts with the goal & works backward, chaining through rules to find known facts to support the goal

Properties → ① It is a Top Down approach

② It is based on Modus Ponens inference rule

③ The goal is broken into subgoals to prove that the facts are true.

④ It is goal driven approach as the list of goals decides which rules are selected & used.

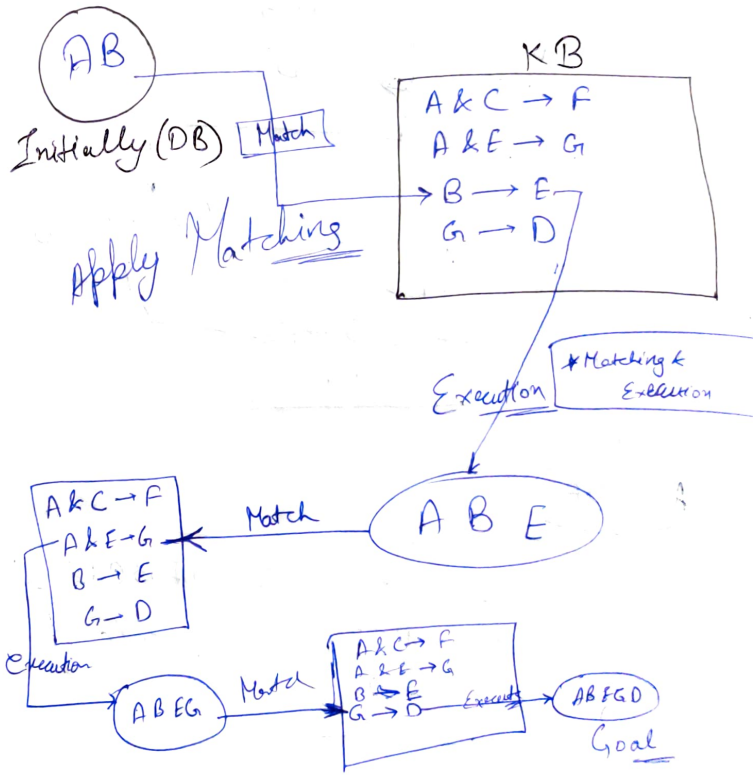
Applications → Game Theory, Automated Theorem Proving tools, Inference Engines, Proof Assistants etc.

Difference b/w

Forward Reasoning	Backward Reasoning
(1) It starts from known facts & applies inference rules to extract more data until it reaches to the goal. (Forward driven)	(1) It starts from the goals & works backward through inference rule to find the required facts that support the goals. (Backward driven)
(2) A Bottom Up Approach	(2) A Top Down Approach
(3) Data Driven as goal is reached using available data	(3) Goal Driven as process starts from goals & divided and into some goals to extract the facts.
(4) Uses BFS strategy.	(4) uses DFS strategy.
(5) It tests for all the available rules.	(5) Only tests few required rules
(6) It is suitable for planning, monitoring, controlling & interpretation applications	(6) Suitable for diagnostics, prescription, & debugging applications.
(7) Can generate infinite no. of possible conclusions	(7) Finite no. of possible conclusions generated
(8) It is aimed for any conclusion	(8) Only aimed for required data

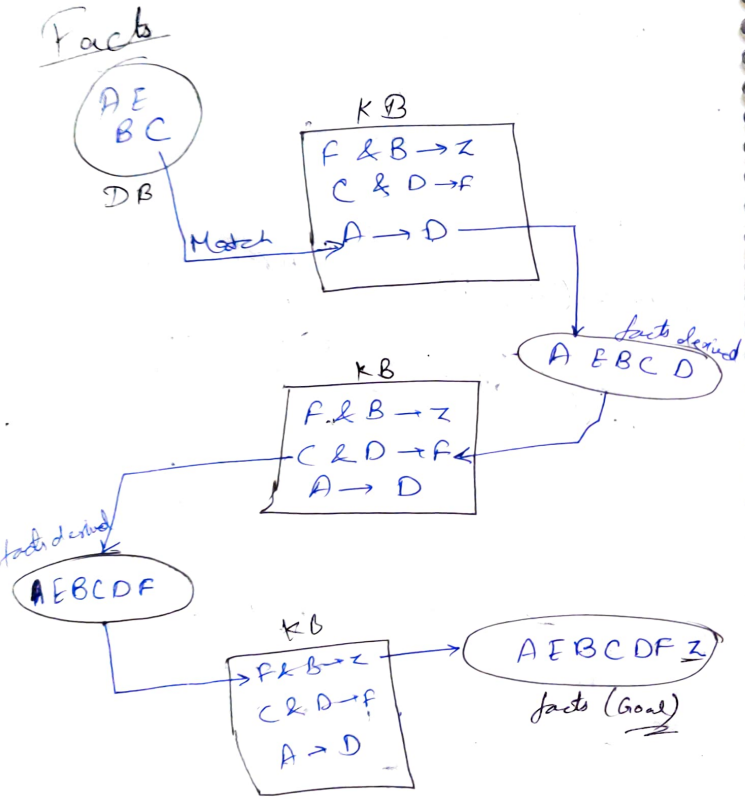
Eg → (1) If A & C then F
(2) If A & E then G
(3) If B then E
(4) If G then D

Prove → If A & B are true then D is true.



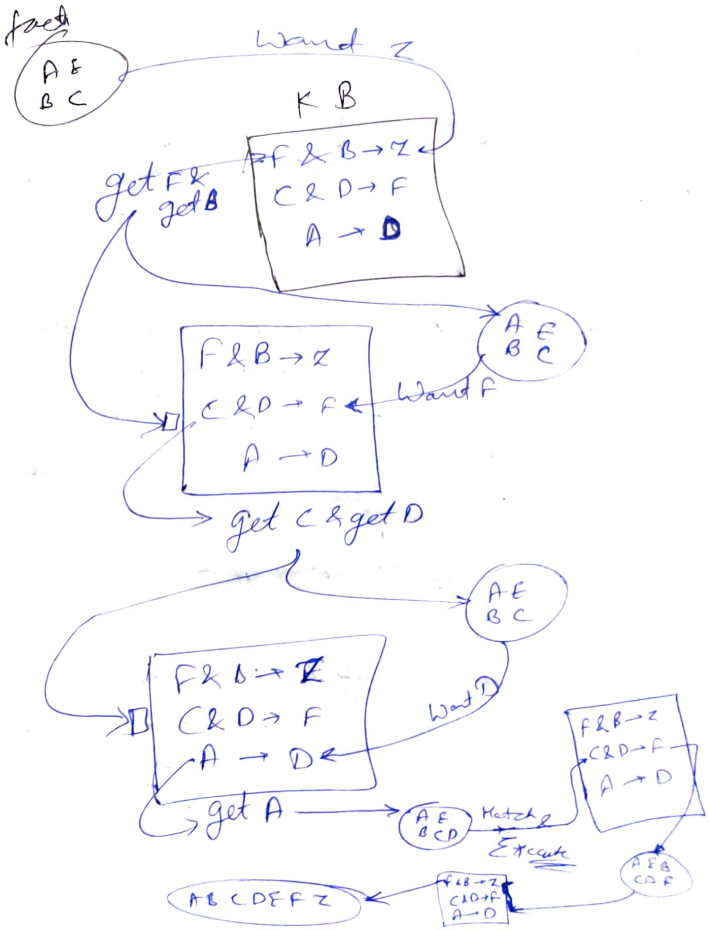
Q) Goal State = Z

Termination Cond^n → Stop if Z is derived.



Q) Goal State: Z

Termination Cond^n → Stop if Z is delivered



26/10/22

Semantic Networks

Ontology

* KR

- Propositional Logic
- FOL
- Rule based System
- Semantic N/w
- Frames

* **Semantic Networks** → It is the graphical notation for representing knowledge in interconnected nodes pattern. It is extremely popular in AI & NLP as it represents knowledge & supports reasoning. It is sometimes considered as an alternative of predicate logics.

Nodes
- Circle (Physical objects)
- Ellipse (Conceptual objects)
- Rectangle (Situational)

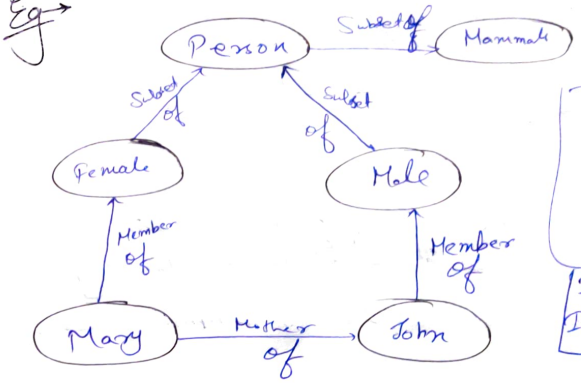
* Arcs represent relationship b/w objects. (Link) Arrows

Link labels specifies relationship. Semantic Network is also known as Associative Nets as they show association b/w nodes.

Components of Semantic Networks

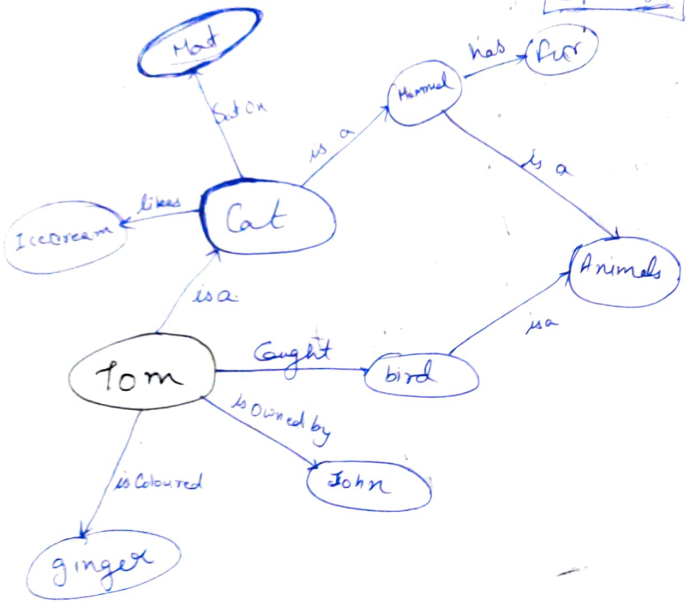
1. Lexical Component → nodes, links, labels.
2. Structural Components → directed links
3. Semantics Components → Definition related to links or nodes (Representation of facts)
4. Procedural Components → Constructor (creation of new links)
   → Destructors (Removal of links)

Eg →



Disadv. of Semantic Net → 1. ISA & HAS RELATIONS have wrong interpretation of knowledge
↳ Lack of Formality

Eg →
- S-1 → Tom is a Cat.
- S-2 → Tom caught a bird
- S-3 → Tom is owned by John
- S-4 → Tom is ginger in color
- S-5 → Cats like icecream
- S-6 → The Cat on the Mat.
- S-7 → Cat is a Mammal
- S-8 → A bird is an animal
- S-9 → All Mammals are animals
- S-10 → Mammals have furs



Frames → It is the collection of attributes or slots & associated values that describes some real world entity.

It uses a data structures called records to represent knowledge in Semantic Network.

Each frames represents a nodes in semantic Network as a class or instance & each relationship is a slot.

Attributes attached with each slot → (1) **Instance**: It relates slot with a class.

(2) Defination: It provides slot def^n or value

(3) Default: Slot default values are given here

(4) Domain: Provides the slot elements domain

(5) Range: Specifies the class of which Elements.

(6) Range Constraints → Logic Expression which must is true

Eg → $(6 < i < 10)$

(7) To Compute → Provides the value of slot that is to be computed.

(8) Single valued → The function returns single value

(9) Inverse → Slot Inverse used in reasoning.

(10) Transfers Through → It decides Inheritance

Reasoning Actions that can be performed using Frames

(1) **Relating the def^n** → It is the propagation of def^n to relate all information

Eg → is a (Inverse)

(2) **Inheritance** → All values (including default values of slot) are Inherited

(3) **Legality of Value** → It checks the legality of slot values

(4) Consistency checks → Verifythe slot value consistency before adding to the frames. (Domain) (Range Constraints)

(5) Maintaining Consistency → when one slot is updated its inverse should also be updated (Inverse)

⑥ Computation of Value of Slot → To Compute & Transfers Process

Eg → Emp Details

Ram Mohan (← Slot

Instance (Profession (Value Manager))
(EmpId (Value 106))
(Address (Value Jaipur))

Slot Value

)

Eg → Tweety is a Yellow Bird having Wings to fly.

Tweety (

(Species (Value Bird))
(Colour (Value Yellow))
(Activity (Value Fly))
(Body Part (Value Wings))

)

→ Weak Slot & Filler Structures
↙ Semantic Networks
↘ Frames

Disadvantage of Frames

The Updations in Frames must necessarily be propagated to the required places in the Semantic Network

★ Strong Slots & Filler Structures

- Script
- Conceptual Dependency.

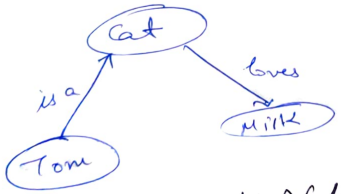
28/10/22

Poulomi Mam

Earlier we need to form the mathematic logic for every sentence.

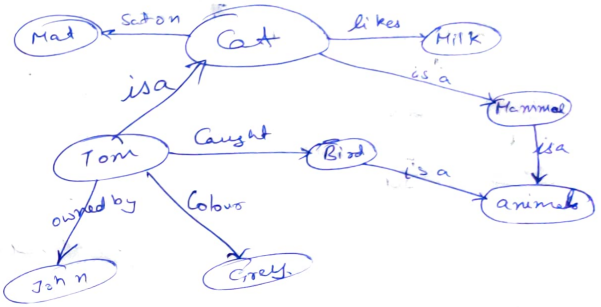
↳ But if we have a technique to maintain it in natural language. → much better & easier to work with

Eg → Tom is a cat.
Cat loves milk.

Logic	Representation
• ∀x Cat(x) loves(x, milk) • Cat(Tom)	

K/a: Semantic Net

Eg → (i) Tom is a cat
(ii) Tom caught a bird
(iii) Tom is owned by John
(iv) Tom is grey in Color
(v) Cats likes Milk
(vi) A Cat Sat on a Mat
(vii) A cat is Mammal
(viii) A bird is a animal
(IX) All mammels are animals

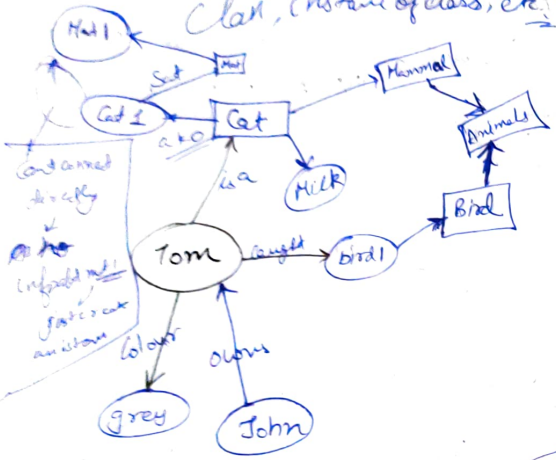


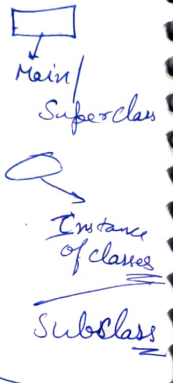
Every node is a class & arcs are relationship among the node.

Disadv → Is Tom a cat that is sitting on the mat. or any another cat

No difference between node of class or no of instance of class.

So, we need something to distinguish b/w Class, instance of class, etc.





for each & every instance → diff. obj node

Not allow a passive sentence

Convert to Active

We have two types of relations

- is a
- a kind of (a-k-o)

Tom is a cat

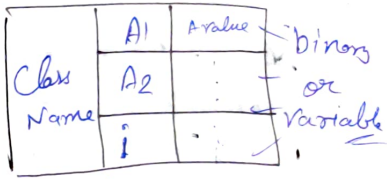
for any further info added → no. of nodes keep inc. for each

So, if many cats killed diff. birds owned by diff owners sitting a diff mats.

Thus to avoid this repetation & redundancy & to save space

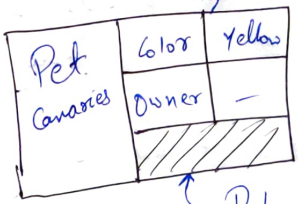
we use Frames

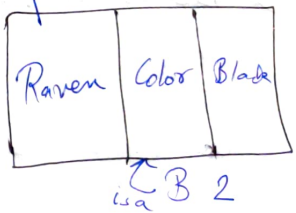
Frames
Slot
Slot Value / Filler



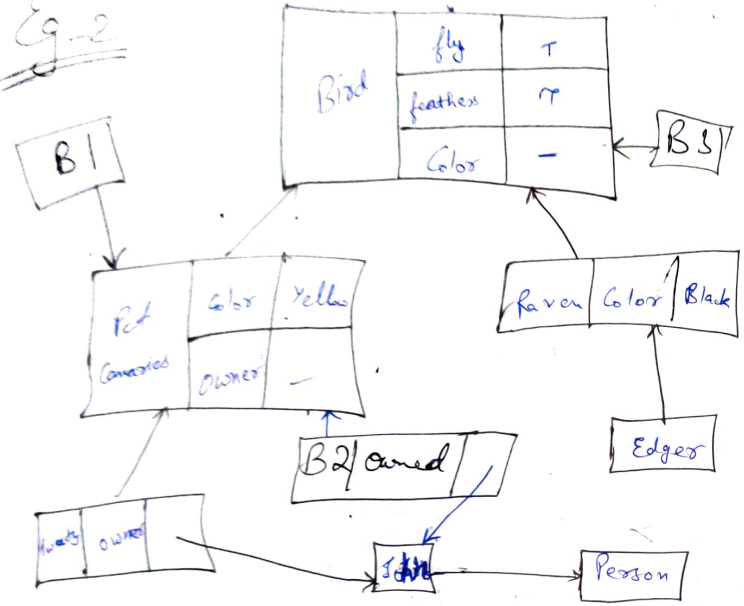
frame

Eg →





Eg-2



Let B1 be a pet canary unowned

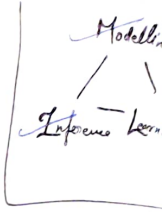
& B3 be a bird other than Pet Canaries & Raven

Creating new relationship is much easier then Semantic Net

* Let B2 be a pet Canary also owned by John

31/10/22

Learning



* Information is prerequisite for Learning

Any detail of state, feature of states, Transition of states, etc by the agent is Information

Eg → This is AI class, Mam is teaching group of students who want to learn abt learning mod of AI.

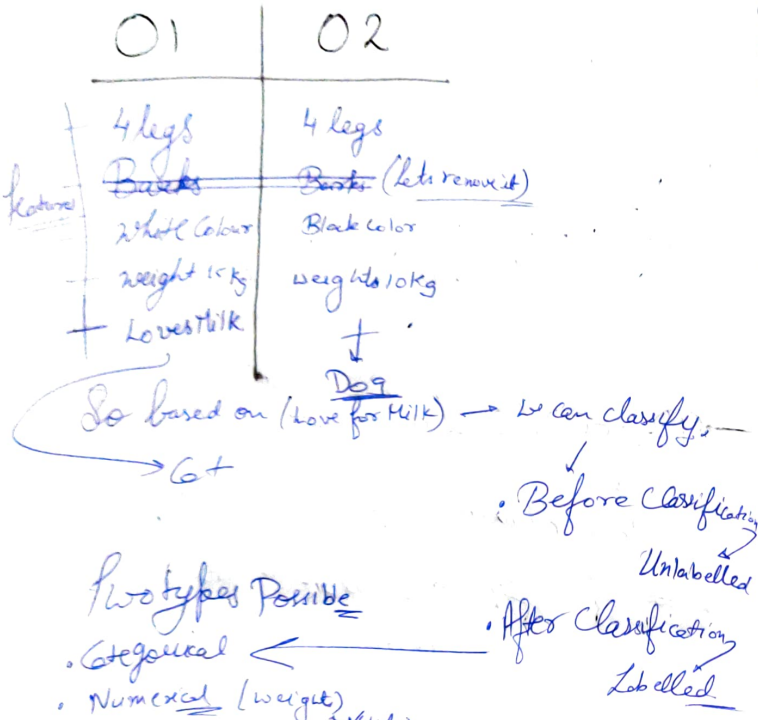
→ Data Vs Information

Compact way of representing the information

* Data is of two type → labelled → Some features are there which can be used to differ entities
  - Unlabelled →

→ Depending of types of data, we have types of Learning Mechanisms

- Supervised Learning (Labelled Data)
- Unsupervised Learning (Unlabelled Data)
- Reinforcement Learning (Not such data available (e.g. Start State, Goal State given. Take action based on rewards)



Supervised Learning

- Predictive ML model. (for any new Data Pt.)
- needs labelled Training Data

* System generates the Model with the feature of the Training Data
* On obtaining the new data pt for prediction, it will get the features & then formulate it based on the existing Model.
- Based on the result of formulation, the Prediction will be made.

~~Types of Regression Models~~ →

Types of Supervised Learning →

- Regression Model → predicts a numerical Value (weight, age, etc)
  - Continuous Interval
  - Eg → House Price Prediction Model, Airport Price Model.
- Classification Model → predicts a State / Category (Cat / dog)
  - Discrete set
  - Eg → Spam Detection Model. [Yes / No OR Spam / Primary Inbox]
    - Whether / Is ?
    - What ?

Unsupervised Learning

* data has no labels (unlabelled Data)
* only features, no target
* Can group the data pts based on similarities → Clusters

→ Also help in supervised by removing the fear of data pts

Types of Unsupervised Learning

- clustering → This is the task of grouping into clusters
- Dimensionality Reduction →

Eg →

01

2 balconies
2 bedrooms
800 sqft
} Size

Near to Road
Nearer to School
Nearer to Hospital
} Area-Quality

So, we have reduced 6 to 2 dimension without the loss of Data generality.

Now, based on these 2 features, we can perform Clustering