Unit - 4

Chapter - 1 [Knowledge Representation]

Introduction; Approaches to Knowledge Representation (Relational Knowledge, Knowledge represented as logic, procedural knowledge); Knowledge representation using semantic network (inheritance in semantic net); Extended semantic networks for KR (inference rules, deduction in extended semantic network, example for illustrating inference methods, inheritance, implementation); Knowledge representation using frames (inheritance in frames, implementation of frame knowledge).

• Introduction:

Traditionally Knowledge Representation was concerned with the ways in which knowledge was stored in an information system to accomplish a given task. In AI KR is an important area because intelligent problem solving can be achieved and simplified by choosing an appropriate knowledge representation technique. Representing knowledge in some special/specific ways makes certain problems easier to solve. AI has a number of representations that are devised to structure information. Fundamental goal of KR is to represent knowledge in a manner that facilitates the process of inferencing (ie drawing conclusions) from it.

/* KR in AI is not just about storing data in a database, it allows a machine to learn from knowledge & behave intelligently like human being */

- KR is a field of AI that focuses on designing computer representations that capture information about the world that can be used for solving complex problems.

- Several programming languages oriented to KR have been developed till date. Example Prolog, KL-one, SGML, XML, RDF, ...
- Any KR system should possess properties such as learning, efficiency in acquisition, representational adequacy, inferential adequacy.

• Properties:

→ Learning refers to a capability that acquires new knowledge, behaviours, understandings etc.,) It is to classify information to avoid redundancy & replication. knowledge is gained by reasoning, logic, experience, observing world, mathematical proofs, scientific methods...,

→ Efficiency in acquisition refers to ability to acquire new knowledge using automatic methods. whenever possible rather than relying on human intervention.

→ Representational adequacy refers to the ability to represent required knowledge.

→ Inferential adequacy refers to ability to manipulating knowledge to produce new knowledge from the existing one.

- Efficiency of a method depends greatly on the representation scheme of the knowledge. Many AI methods have tried to homodel human intelligence. Knowledge representation

Scanned with CamScanner

is a core component of a number of applications such as expert systems, machine translation system, computer-aided maintenance system, information retrieval system & database front-ends.

- In expert systems there are various ways of representing knowledge namely predicate logic, semantic networks, frames, conceptual dependency etc., which have originated from theories of human information processing.
- In predicate logic, knowledge is represented in the form of rules & facts. An expert system that incorporates knowledge in the form of rules & facts is called as rule based expert system.

- **Approaches to Knowledge Representation:**

AI program use structures known as knowledge structure to represent objects, facts, relationships, & procedures. The main function of these knowledge structures is to provide expertise & information so that program can operate in an intelligent way. Knowledge structures are usually composed of both traditional & complex structures such as semantic network, frames, scripts, conceptual dependency structure ..., . A knowledge base is a special type of database that holds knowledge of the domain. It stores, updates & retrieves information. Organized in hierarchical structure with in-built inheritance & influencing capabilities. Some KR schemes are:

- **Relational Knowledge:**

→ Comprises objects consisting of attributes & associated values. It is simplest way of storing facts.
→ Each fact is stored a row. A table is defined as a set of data values that is organized using a model of horizontal rows & vertical columns.
→ The columns are identified by attribute names, which the rows are identified by corresponding values appearing in a particular column subset.
→ This representation helps in storing the facts but gives little opportunity for inferencing; however such structure can be used by inference engine.

Relational Table

Name	Age (in years)	Sex	Qualification	Salary in rupees
John	38	Male	Graduate	20000
Mike	25	Male	Undergraduate	15000
Mary	30	Female	PhD	25000
James	29	Male	Graduate	18000

- He can easily obtain answers from queries like example.
  - ↳ What is age of John?
  - ↳ How much does Mary earn?
  - ↳ What is qualification of Mike?

Scanned with CamScanner

- **Knowledge Represented as Logic:**

Inferential capability can be achieved if knowledge is represented in the form of formal logic.

Eg: All humans are mortals cannot be represented using relational approach; instead it can be easily represented in predicate logic as

$(\forall x)$ human$(x) \leftarrow$ mortal(x)

If we have a fact John is human, then we can easily infer that John is mortal. The advantages of this approach are that we can represent a set of rules, derive more facts, thruths, & verify the correctness of new statements.

- **Procedural knowledge:**

→ Procedural knowledge is encoded in the form of procedures which carry out specific tasks based on relevant knowledge.

→ The advantages of this approach are that domain specific knowledge can be easily represented and side effects of actions may also be modelled.

→ But this is a problem of completeness (all cases may not be represented) & consistency (all deductions may not be correct.

/* Knowledge organized in the form of semantic n/w & frames is refered to as inherita -ble knowledge */

- **Knowledge Representation using Semantic Network:**

→ The basic idea is to derive meaning of a concept from its relationship with other concepts, an the information is stored by interconnecting nodes with labelled arcs.

Eg: Every human, animal & birds are living things who can breathe & eat. All birds can fly. Every man & Woman are human who have two legs. A cat has fur & is an animal. All animals have skin & can move. A giraffe is an animal & has long legs & is tall. A parrot is a bird & is green in colour. Jch is a man.

→ He can represent such ~~semantic structure~~ knowlege using a structure called a semantic networks (semantic net). It is conviniently represented in a graphical notation where nodes represent concepts (or) objects & arcs represent relation between two concepts.

→ It can be described as a directed graph consisting of vertices representing concepts & edges that represent semantic relations between the concept. Nodes can be easily added in this network.

→ Concepts (or) objects are related to each other by certain relations. These relations are represented by bold directed links. They may be defined as:

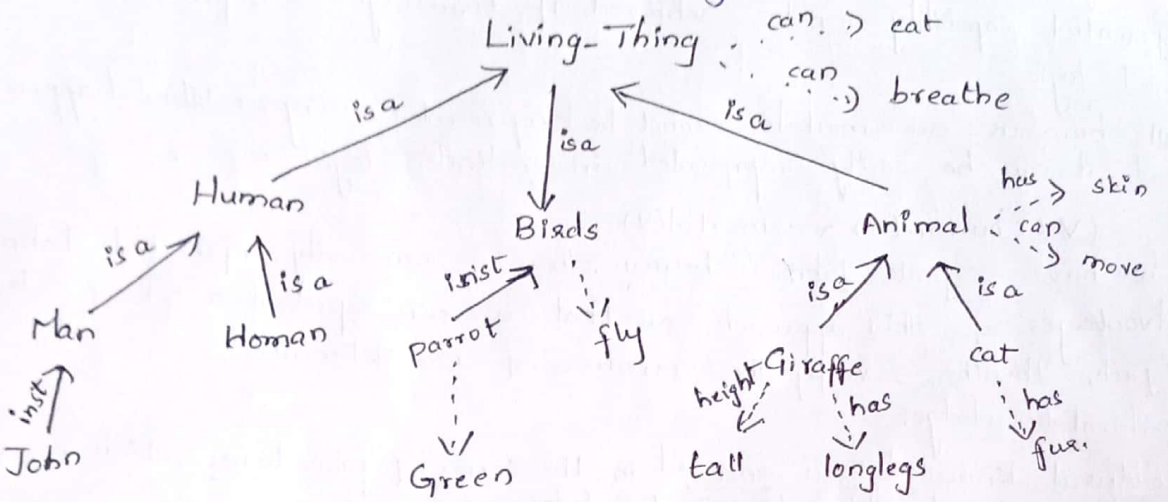
is a – relation connect two classes (generic concept), when one concept is a kind of subclass of the other concept. Eg: Man *is a* human
(Man: subclass, human: class)

inst – relation relates specific members of a class. Eg: John is an instance of Man

Scanned with CamScanner

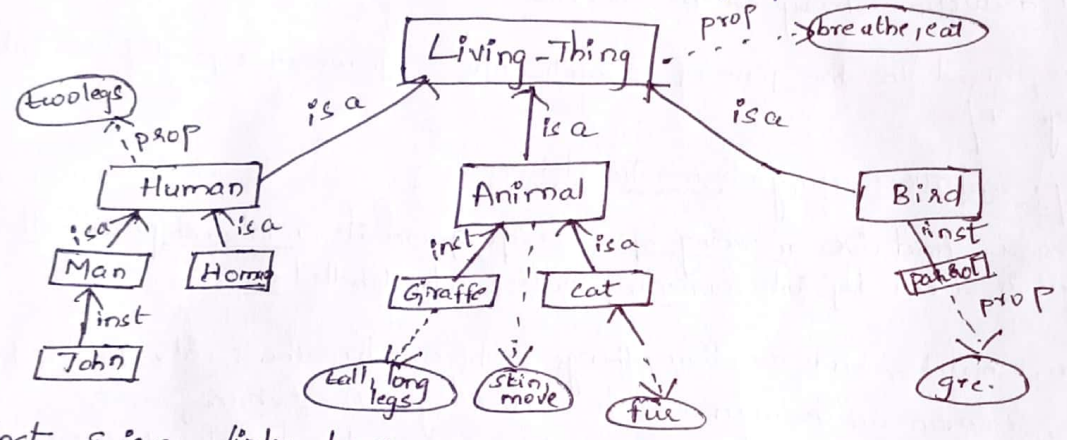
→ Properties attached to a particular concept can be easily inherited by

Knowledge Representation using Semantic Net



→ Other relations such as {can, has, colour, height} are known as property relations, represented using dotted lines pointing from concept to its property.

Concepts Connected with prop Links



→ inst & is a links have well defined meaning, whereas other links which connect the attributes of class vary in their interpretations.

→ The semantic net interpreter cannot understand all attributelinks unless their semantics are encoded into it. Since all attributes are properties we can designate a unique name to such relations say prop link.

→ The interpretation of prop link will be simple that the attributes attached to a class is one of its properties.

→ The square nodes represents concepts/objects connected with is a (or) inst links, whereas oval nodes represent property attributes attached to square nodes.

→ Property links are shown as dotted lines for more clarity.

- **Inheritance in Semantic Net:**

→ Hierarchical structure of knowledge representation allows knowledge to be stored at the highest possible level of abstraction which reduces size of Knowledge Base (KB)

→ It is a natural tool for representing taxonomically structured information associated with nodes in semantic net. & ensures that all the members and sub concepts of a concept share common properties.

→ Maintain consistency of KB by adding new concepts & members to existing ones.

Scanned with CamScanner

→ Properties attached to a particular concept can be easily inherited by all subconcepts & their members.

The algorithm that retrieve a property of an object (concept/instance) is

Property inheritance Algorithm

input: Object & property to be found from Semantic Net
output: Returns yes, if the object has the desired property else returns false;
Procedure:

- find an object in semantic net;
- found = false;
- While [(object ≠ root) OR found] DO
  {
  - if there is an attribute attached with an object then Found = true;
  else {object = is a (object.class) or object = inst (object, class)}
  };
- If found = true then report 'Yes' else report 'No';

→ Semantic net can be implemented in any programming language along with an inheritance procedure implemented explicitly in that language.

→ Prolog language is very convenient for representing an entire semantic structure in form of facts (relation as predicate & nodes as arguments) and inheritance rules.

→ Inheritance is easily achieved by unification of appropriate arguments in Prolog.

- **Prolog Facts**

Is a facts	Instance facts	Property facts
is a (living_thing, nil) is a (human, living_thing) is a (animals, living_thing) is a (birds, living_thing) is a (man, human) is a (woman, human) is a (cat, animal)	inst (john, man) inst (giraffe, animal) inst (parrot, bird)	prop (breathe, living_thing) prop (eat, living_thing) prop (two_legs, human) prop (skin, animal) prop (move, animal) prop (fur, bird) prop (tall, giroffe) prop (long_legs, giroffe). prop (tall, animal) prop (green,

- **Inheritance Rules in Prolog:**

→ In class hierarchy structure, a member subclass of a class is also a member of all super classes connected through isa link.

→ A property of a class can be inherited by lower subclasses.

→ Simple prolog rule for handling inheritance is as follows & various queries can be answered by following inheritance program.

Scanned with CamScanner

in the sentence john gave an apple to mike in the kitchen, it is easy to add location(ε, kitchen) to the set of facts

Instance Rules

```
instance(X, Y)   :-  inst(X, Y).
instance(X, Y)   :-  inst(X, Z), subclass(Z, Y).
```

Subclass Rules

```
subclass(X, Y)   :-  isa(X, Y).
subclass(X, Y)   :-  isa(X, Z), subclass(Z, Y).
```

Property Rules

```
property(X, Y)   :-  prop(X, Y).
property(X, Y)   :-  instance(Y, Z), property(X, Z).
property(X, Y)   :-  subclass(Y, Z), property(X, Z).
```

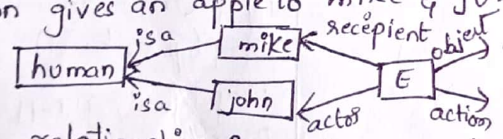
Various Queries for Inheritance program

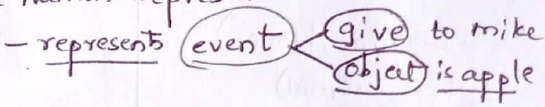
English Query	Prolog goal	Output
Is john Human?	?- instance(john, human).	Yes
Is parrot is living thing?	?- instance(parrot, living-thing).	Yes
Is giraffe an animal?	?- instance(giraffe, animal).	Yes
Is Human is subclass of living thing?	?- subclass(Human, living-thing).	Yes
Does parrot fly?	?- property(fly, parrot).	Yes
Does parrot have fur?	?- Does parrot have fur?	No
Does John breathe?	?- property(john, breathe).	Yes
Does cat fly?	?- property(fly, cat).	No

- **Extended Semantic Networks for KR:**

Logic & semantic networks are two different formalisms that can be used for KR. Simple semantic n/w is represented as a directed graph whose nodes represents concepts/objects & arcs represent relationships between concepts/objects. It can also express collections of variable free assertions.

Eg: john gives an apple to mike & john & mike are human represented in semantic net as





The relationships in n/w can be expressed in clausal form of logic as follows

```
object(ε, apple).
action(ε, give).
actor(ε, john).
recipient(ε, mike).
isa(john, human).
isa(mike, human).
```

- Predicate relations corresponding to labels on the arcs of semantic networks always have two arguments. Therefore, the entire semantic net can be coded using binary representation (two-argument representation). Such representation is advantageous when additional information is added to a given system. For example.

Scanned with CamScanner

in the sentence john gave an apple to mike in the kitchen, it is easy to add location(E, kitchen) to the set of facts.

- In first order predicate logic, predicate relation can have n arguments where $n \geq 1$. The predicate logic representation has greater advantages compared to semantic net representation as it can express general propositions in addition to simple assertions. Eg; John gives an apple to everyone he likes

give(john, x, apple) ← likes(john, x).

variable — logical connective implied by — Right side conditions(s)

left hand contains conclusion(s)

- Despite all advantages, it is not convinient to add new information in an n-ary representation of predicate logic
- A clause in logic can have several conditions, all of which must hold for the conclusion to be true & can have several alternative conclusions, atleast one of which must hold if all the conditions are true.

Eg: ① if john gives something he likes to a person, then he also likes that person can be expressed in casual logic as

likes(john, x) ← gives(john, x, y), likes(john, y).

② every human is either male or female is expressed as

male(x), female(y) ← human(x)

- In conventional semantic network we cannot express casual form of logic. For this R Kowalski & his colleagues (1979) proposed an extension/extended Semantic network (ESNet) that combines advantages of both logic & semantic n/ws
- ESNet can be interpreted as variant syntax for the clausal form of logic. it has same expressive power as that of predicate logic with well defined semantics, inference rules, & a procedural interpretation.
- It incorporates advantages of using binary relation as semantic n/w rather than n-ary relations of logic. ESNet is much powerful representation compared to logic & Semantic n/w.
- In ESNet, terms are represented by nodes similar to as done in conventional Semantic network; constant, variable, & functional terms are represented by constant, variable & functional nodes respectively.
- Binary predicate symbols in casual logic are represented by labels on arcs of ESNet. An atom of the form love(john, mary) is an arc labelled as love with two end nodes representing john & mary. The direction of arc (link) indicates order of arguments of predicate symbol which labels arcs as follows

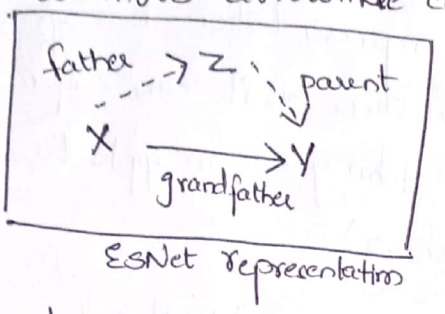
john —love→ mary

time

Scanned with CamScanner

Conclusions & Conditions of clausal form are represented in ESNet by different kinds of arcs. The arcs denoting conditions (negative atoms) are drawn with dotted arrow lines(- - ->), while the arcs are denoting conclusions (positive atom) are drawn with continous arrowlikee called assertion links (→)

clausal representation | Rule



ESNet representation

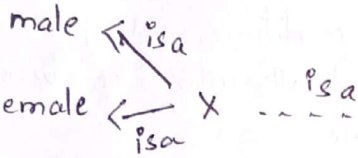
grandfather(X,Y) ← father(X,Z), parent(Z,Y)

X, Y – Variables

grandfather(X,Y) is consequent (conclusion) & father(X,Z) and parent(Z,Y) are antecedents (conditions).

Similarly clausal rule male(X), female(X) ← human(X) can be represented using binary representation as

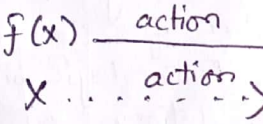
isa(X, male), isa(X, female) ← isa(X, human). and subsequently in ESNet as



- **Inference Rules:**

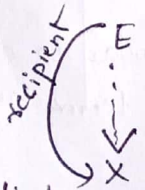
Inference rules are embedded in the representation itself. Some of the inference rules are:

- The representation of the inference for every action of giving, there is an action of taking in clasual logic is action(f(X), take) ← action(X, give). The intepretation of this rule is that the event of taking action is a function of the event of giving action. In the ESNet representation, functional terms, such as f(X), are represented by a single node. The representation of the statement action(f(X), take) ← actim(X, give) in ESNet is



- The inference rule that an actor who performs a taking action is also the recipent of this action & can be easily represented in clausal logic, ESNet as

recipient(E,X) ← action(E, take), actor(E,X).



The hierarchy links, isa, inst & part-of (or prop), are available in semantic networks and also available as special cases in ESNet. Part-of-link has a hidden existential quantifier.

Eg: part of link is

two-legs —part-of→ human

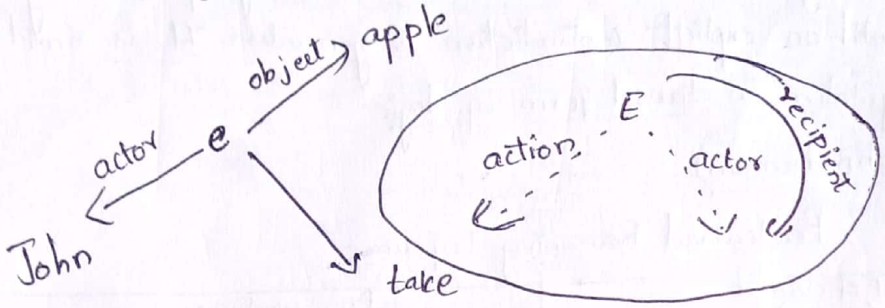
interpretation is → for every human, there exists two legs is part of that human

Scanned with CamScanner

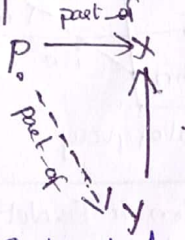
Eg: Taking actions

recipient(E, X) ← action(E, take), actor(E, X)
object(e, apple).
action(e, take).
actor(e, john).

e is an actual event. Clausal rule is enclosed in ellipse



- The contradiction in ESNet can be represented as P part-of X is conclusion and P part-of Y is condition, where Y is linked with X via isa link. Such kind of representation is contradictory & hence there is contradiction in ESNet



—> Contradiction in ESNet.

- Deduction in Extended Semantic Networks:

Inference rules along with its semantics form a part of ESNet. In logic there are two types of inference mechanism, namely

→ forward reasoning inference mechanism (bottom-up approach)
→ backward reasoning inference mechanism (top-down approach)

* Forward Reasoning Inference:

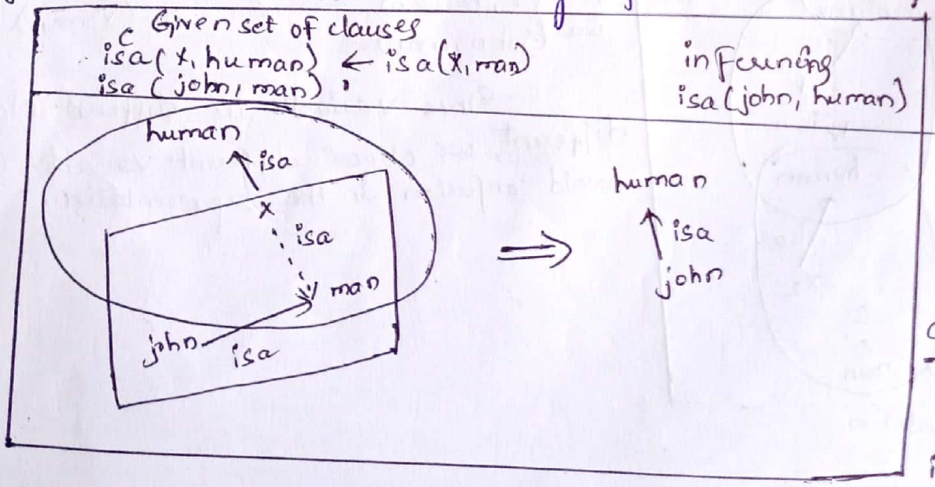
Given an ESNet, apply the following reduction (resolution) using modus ponen rule of logic {i.e given (A ← B) and B, then conclude A}.

Eg: isa(X, human) ← isa(X, man)
isa(john, man).

Using MP rule we can easily derive isa(john, human) hold true

isa(john, human) can be derived / inferenced using ESNet

Forward Reasoning Inference Mechanism



Contradiction enclosed in rectangular box.

The new assertion isa(john, human) is inferred by elimination of assertion & their denial links

isa(X, human) — assertion link
isa(X, man) — denial link

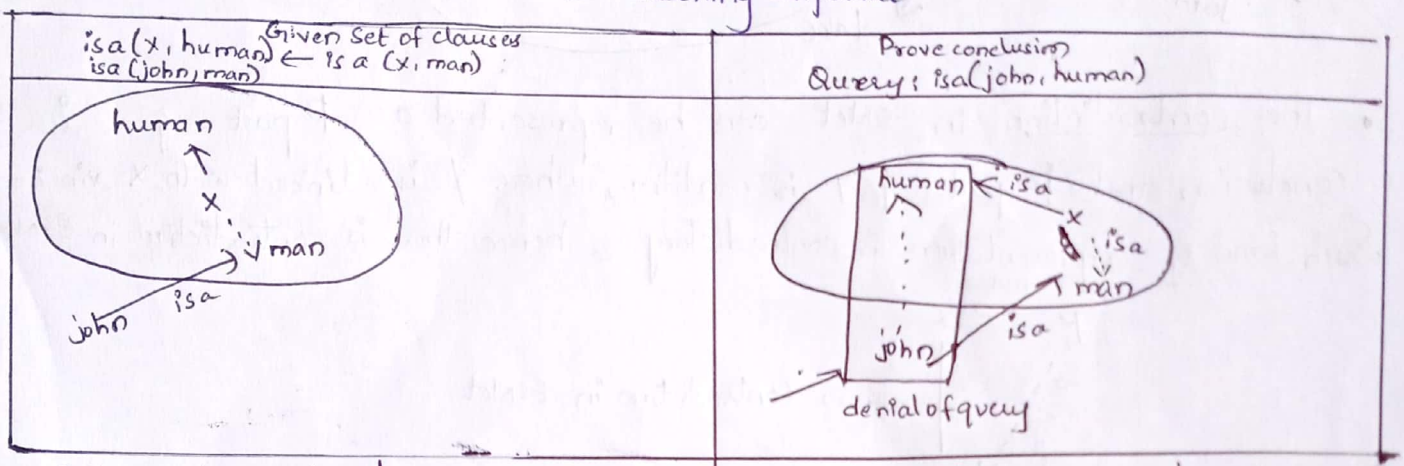
Scanned with CamScanner

* Backward Reasoning Inference:

In backward inferencing mechanism, we can prove a conclusion or goal from a given ESNet by adding denial of conclusion to network and show that resulting set of clauses in network gives contradiction. This is done by performing successive steps of resolution until an explicit contradiction is generated. It is similar to proof by resolution refutation in clausal form of logic

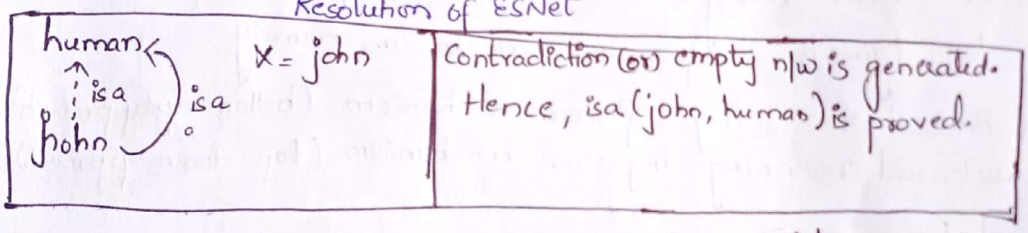
Eg: To prove isa(john, human)

Backward Reasoning Inference



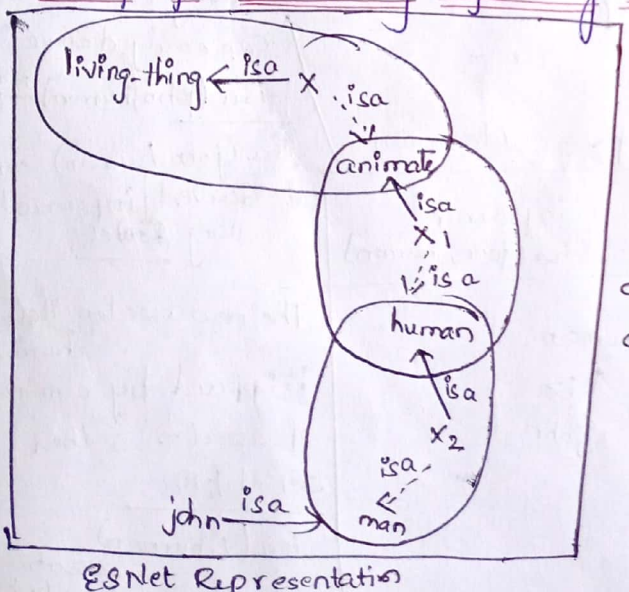
After adding denial link in ESNet, we get the reduction in ESNet as shown below, by elimination of assertion & their denial link with the help of appropriate substitution

Resolution of ESNet



Thus inferencing & inheritance are important and embedded capabilities of ESNet.

- Example for Illustrating Inferencing Methods:



ESNet Representation

Given clausal form

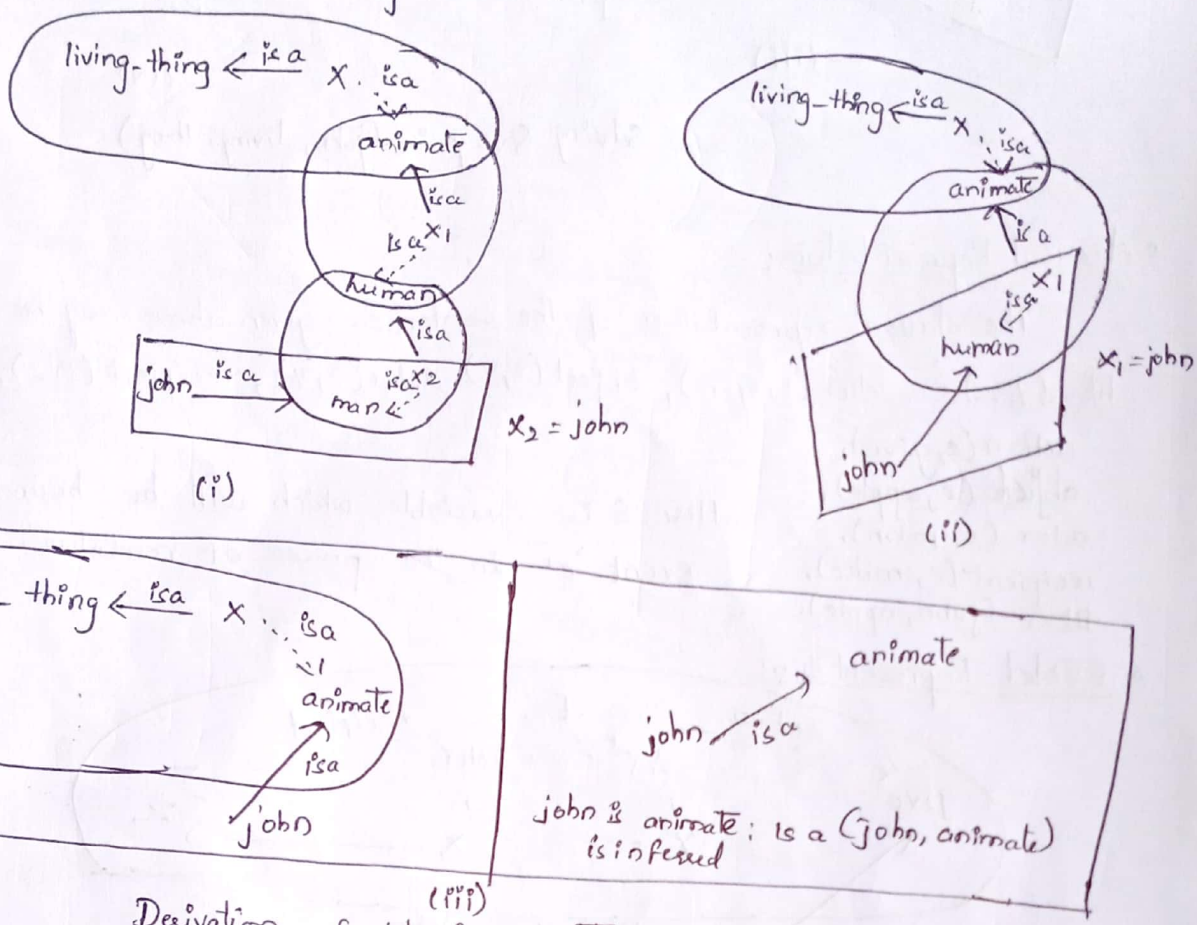
isa(x, living_thing) ← isa(x, animate)
isa(x, animate) ← isa(x, human)
isa(x, human) ← isa(x, man)
isa(john, man)

Since variables in different clauses are different, we choose different variables in ESNet to avoid confusion in the representation.

Scanned with CamScanner

- Forward Reasoning inference:

The new assertion john is human can be inferred (or derived in previous example. The steps for inferening that john is animate is given below. The assertion john isa living-thing can be inferred similarly.

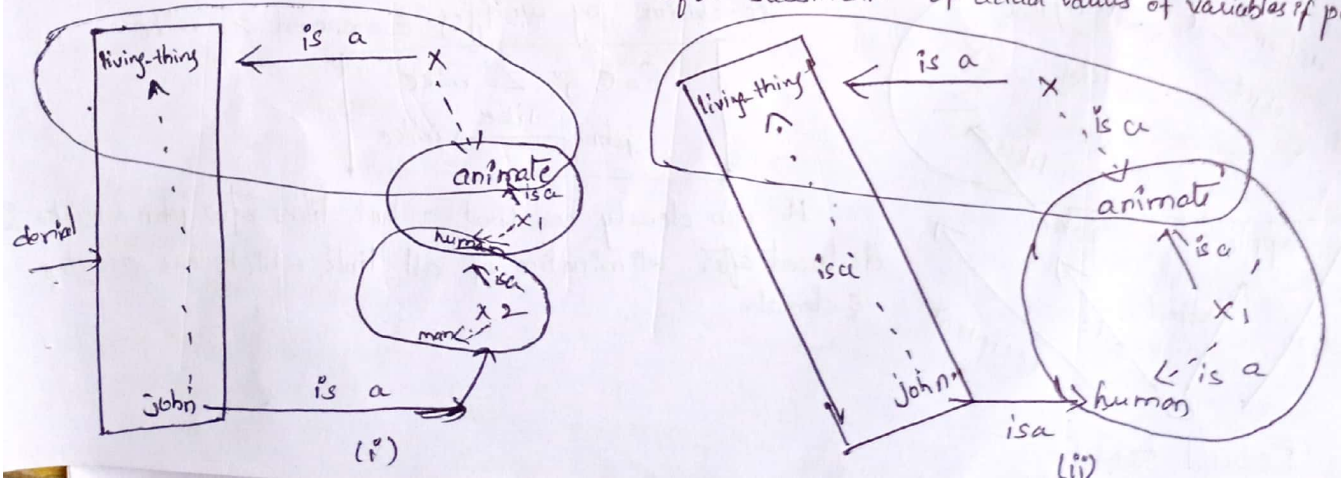


Derivation of john is animate

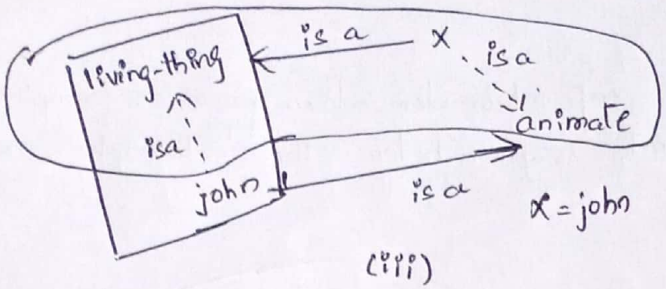
John is living-thing can be derived by binding x with John

- Backward Reasoning Inference:

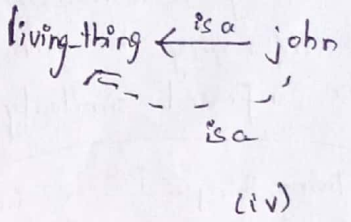
To Solve a query isa(john, living-thing) (i.e, Is John a living thing?) using backward inferening mechanism. In this mechanism, we add the denial of the fact that John is not a living-thing in ESNet & see if we can derive inconsistency (or) an empty network. in following figure each part shows reduction in the n/w by elimination of assertions & their denial links with the help of appropriate substitutions. Denial links are enclosed in rectangular box to distinguish it from rest of n/w. Dotted lines are denial links, while continous lines are assertion links. These are removed after substitution of actual values of variables if possible.



Scanned with CamScanner



(iii)



leads to empty clause (or) contradiction

(iv)

Solving Query is a (john, living-thing)

- **Clausal Representation:**

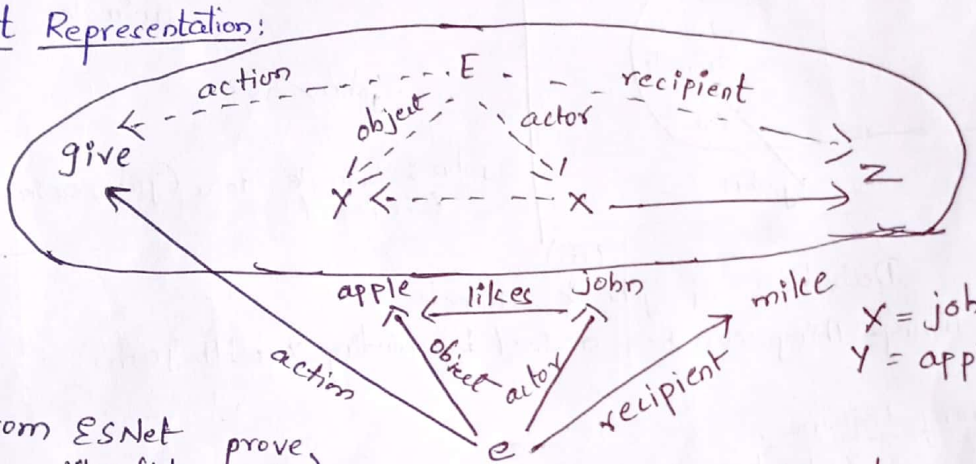
The clausal representation of the sentences given above may be written as

likes(x, z) ← action(ε, give), object(ε, y), actor(ε, x), recipient(ε, z), likes(x, y)

action (e, give).
object (e, apple).
actor (e, john).
recipient (e, mike).
likes (john, apple).

Here ε is a variable which will be bound to an actual event 'e' in the process of resolution.

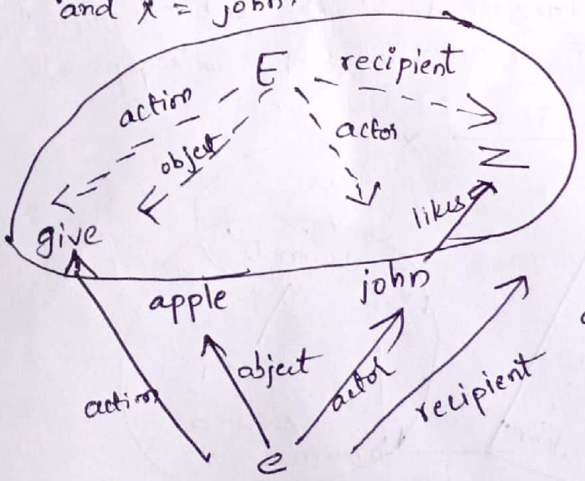
- **ESNet Representation:**



From ESNet prove likes (john, mike) —— done using both < forward reasoning inference / backward reasoning inference

- **Forward Reasoning inference:**

Reduce the n/w by resolving clauses till we obtain the desired assertion. It is easy to eliminate assertion & its corresponding denial with likes links by unifying y = apple and x = john.



Reduced ESNet.

→ The n/w is futher reduced to following conclusion by unifying ε = e and z = mike

ε = e , z = mike
john —like→ mike

He can clearly see that a link likes b/w john & mike is deduced after elimination of all links which are assertns & denials

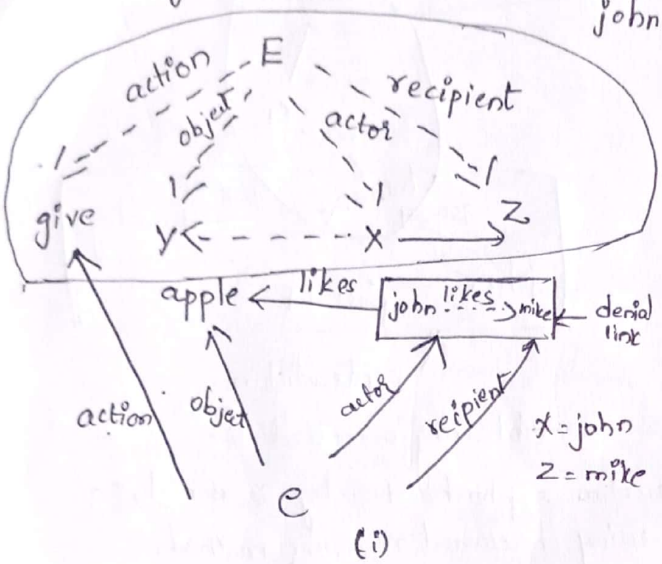
after elimination of assertion

Scanned with CamScanner

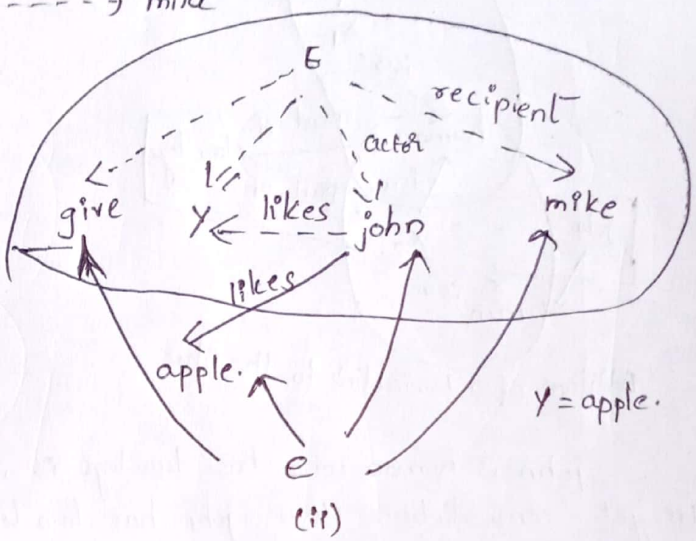
∘ Backward Reasoning inference:

Prove likes(john, mike) using Backward Reasoning method. In order to prove likes(john, mike), add the denial link to the network and try to reduce the network to empty. The denial link is

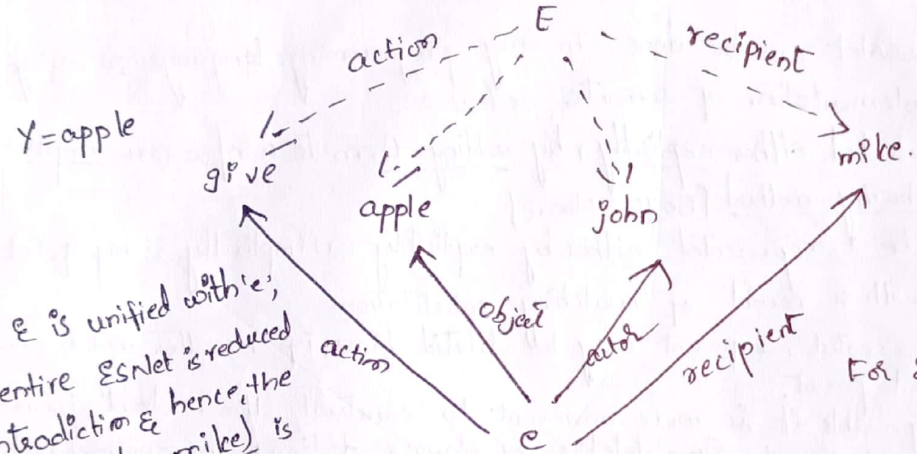
john ----- likes -----> mike



(i)



(ii)



(iii) Reduction to Empty ESNet

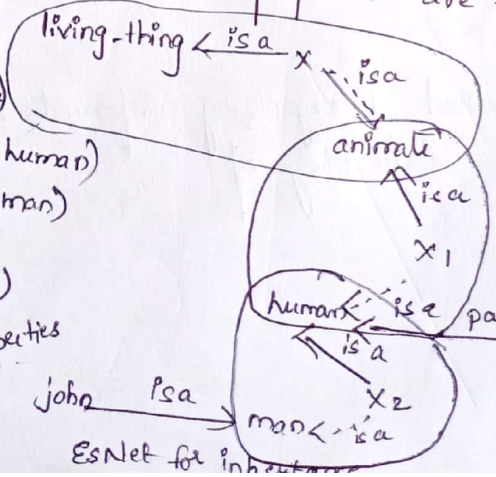
When ε is unified with 'e', then entire ESNet is reduced to contradiction & hence, the query likes(john, mike) is proved using backward reasoning inference process

For ε = e, we get empty network.

∘ Inheritance:

In conventional semantic network, lower level nodes in isa hierarchy inherit properties from higher level nodes unless the properties are redefined in the node itself. Consider

isa(x, livingthing) ← isa(x, animate)
isa(x, animate) ← isa(x, human)
isa(x, human) ← isa(x, man)
isa(john, man)
part-of(human, two-legs)
/* john inherits the properties of having two-legs from human */

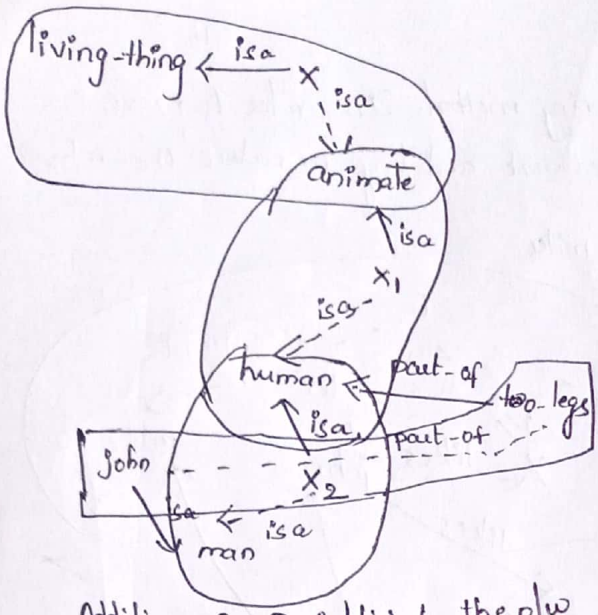


ESNet for inheritance

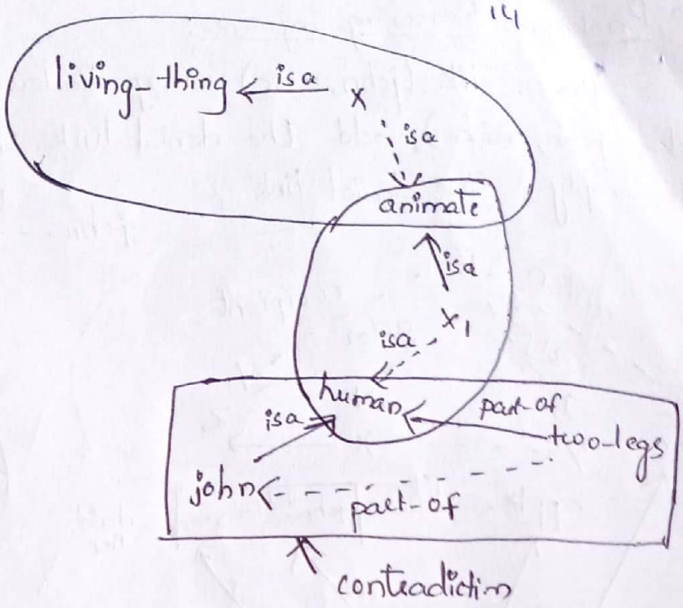
in order to show John has two legs we add denial link of john has two legs to n/w. & we use general purpose reasoning ~~interface~~ inference mechanism & try to get a contradiction.

we get x_2 = john after elimination of assertion & its denial links. The reduced ESNet is

Scanned with CamScanner



Addition of a Denial link to the n/w



Obtaining a contradiction

john is human who has two-legs is a assertion & john has two_legs is denial, so He get contradiction. Thus john has two legs using backward reasoning method.

- **Implementation:**
  - ↳ implementation of ESNet can be done in any programming language (OS) using a tool which facilitates implementation of semantic networks.
  - ↳ clauses can be represented either explicitly, by adding them to a n/w (or) implicitly, by using the structure-sharing method [Boyer & Moore]
  - ↳ The resolvents may be represented either by explicitly (or) implicitly & by pointers to their parents along with a record of matching substitution.
  - ↳ When a resolvent is created, a parent may be delitid from if no other match exists for the atom being matched in that parent.
  - ↳ Whenever deletion is possible it is more convinient to construct the resolvent from the constituents of the deleted parent. Since deletion of clauses destroys the original net, it is advised to save a copy of the original n/w (or) restore it to its original form.

The major point of difference between convention & extended semantic n/ws is as follows
In CSNs procedures are generally written in the host programming language, but in ESNets, procedures are integrated with the rest of database & are executed by same general purpose mechanism which performs inference in the n/w.

- ↳ The ESNet can also be considered as an abstract data structure for the implementation of a proof procedure.
- ↳ A characteristic procedure of feature of ESNet is indexing on argument or predicate where direct access is provided to all atoms (both conditions & conclusions) containing the given term (or) predicate.

/* Indexing scheme is a method of organizing information for effivient auss of the same information at a late stage */

Scanned with CamScanner

Knowledge Representation using Frames:

↳ Frame System used for process of representation of knowledge was first introduced by Marvin Minsky.

↳ Frames are regarded as an extension to semantic nets; each node of semantic net is represented by a frame.

↳ A frame may be defined as a datastructure that is used for representing a stereotyped situation. It consists of a collection of attributes (or) slots & associated values that describe some real-world entity.

↳ There are several types of information attached to each frame. Frame system form a powerful way of encoding information that support reasoning.

↳ With the increase in complexity of a problem, the representation should become more structured for more beneficial use.

↳ Frames may be considered to represent the ways of organising as well as packing knowledge in a more structured form.

↳ Frames are slightly similar to the concept of class/object oriented paradigm; class also contains attributes and methods.

↳ In frames, it consists of attributes (or) slots; slots are described with attribute-value pairs <slot-name, value>. Slots are generally complex structures that have facets/fillers describing their properties.

↳ The values of a slot may be primitive, such as testing string, constant (or) an integer, (or) it may be another frame. Slots may contain value, refer to other frames (relations) (or) contain methods. Most of the frame systems allow multiple values for slots & some systems support procedural attachment as well.

↳ These procedural attachments (methods) can be used for computing the slot value whenever required. Frames may contain triggers for checking consistency (or) obtaining updates of other slots.

↳ Frames are basically a machine usable formalization of concepts of schemata.

General structure of Frame

frame name
slot-fillee
default values
constraints on values within the slots of a frame
pointers (links) to other frames
ako (a-kind-of or subclass
inst (instance)
instantiation procedure
inheritance procedure
default inference procedure
triggers

A frame may contain as many <slots-fillers> pairs as required to describe an object. Fillers are also known as facets. Each slot may contain one/more facets (or) fillers.

Scanned with CamScanner

Lists of Facets in a Frame

Facet Name	Description
value →	value of the slot
default →	default value of the slot & it may be redefined by lower classes
range →	the range of integer or enumerated values that a slot can have
demons →	procedural attachments such as if-needed, if-deleted, if-added, etc.
others →	may contain rules, other frames, semantic net, or any type of other information

A class frame generally has certain default values which can be redefined at lower levels. In case a class frame possesses an actual value facet, then decendent frames cannot modify that value; this value then remains unchanged for all subclasses & instances of that class. The related frames are linked together into frame systems & are organized into hierarchies (or) n/w of frames.

Eg: hospital frame

Slot Name	Facet Name	Facet Value
F-name	value	Metro hospital
Country	default	India
Phone-No	default	23456778
Address	default	abc

Frames in a n/w of frames are connected using the links below. Other required information may be made available using slot-value pair concept

- **ako** — link connects two class frames, one of which is a kind of the other class. A class can define its own slots & also inherits slot-value pairs from its super class. It gives a sub-typing hierarchy where all instances of class frame are instances of super class frames. With this link knowledge representation becomes more structured & memory efficient.
- **inst** — This link connect a particular instance frame to a class frame. An instance class possesses the same structure as a class frame.
- **part-of** — This link connect two class frames one of which is contained in other class.

Example: Descriptions of frames of n/w (Hospital)

* Hospital Frame (Root of the Network)

F-name: hospital
Country: (value-India)
Phn-No: (default-23456778)
Address: (default-NewDelhi)
Director: (default-xyz)
Labs: lab (Lab Frame)
Wards: ward (Ward Frame)
Doctors: director (Doctor Frame)

* Child Hospital Frame

F-name : child_hospital
Ako : hospital (Hospital Frame)
Age : (range-[0-12], (rule-if-added)

* Heart Hospital Frame

F-name ; heart_hospital
Ako : hospital (Hospital Frame)

Scanned with CamScanner

* Lab Frame:

F_name : lab
Part_of : hospital (Hospital Frame)
Pathology : pathology (Pathology frame)
X_Ray : X-ray (X-ray frame)

* Hard Frame

Ename : ward
Part-of : hospital (Hospital frame)
Ortho : orthopedic (Orthopaedic Ward Frame)

* Doctor Frame

F_name : doctor
Part_of : hospital
Qualification : (default-MBBS)

* Orthopedic Hard Frame

F_name : Orthopeadic
Ako : ward (Ward Frame)

* Pathology Frame

F-name : pathology
Part-of : lab (lab-frame)
incharge : (default-xyz)
Types-of test (---)

* X ray frame

F-name : X_ray
Part-of : lab (lab Frame)
Incharge : (default-pqr)

o Frame Instance Descriptions:

o Hospital Frame instance:

F_name : AIIMS
Inst : Hospital Frame
Phn_No : (value-9111659425)
Address : (value-Central-Delhi)
City : (value-New Delhi)
Director : (value-Dr.Smith)
Labs : (lab-name)
Wards : Wards Frame Inst
Doctors : Doctors Frame Inst

o Labs Frame Instance

F-name : lab_name
Inst : lab (lab Frame)
Part-of : AIIMS (Hospital Frame instance)
Pathology : pathology frame instance
X_Ray : X-ray Frame instance.

o Pathology Frame instance:

F-name : pathology-lab-name
Inst : pathology (pathology frame)
Part-of : lab-names (labs frame Inst)
incharge : (value-Mr John)
Types of tests : (value-[ECG, Blood Test, Urine])

o Child Hospital frame instance

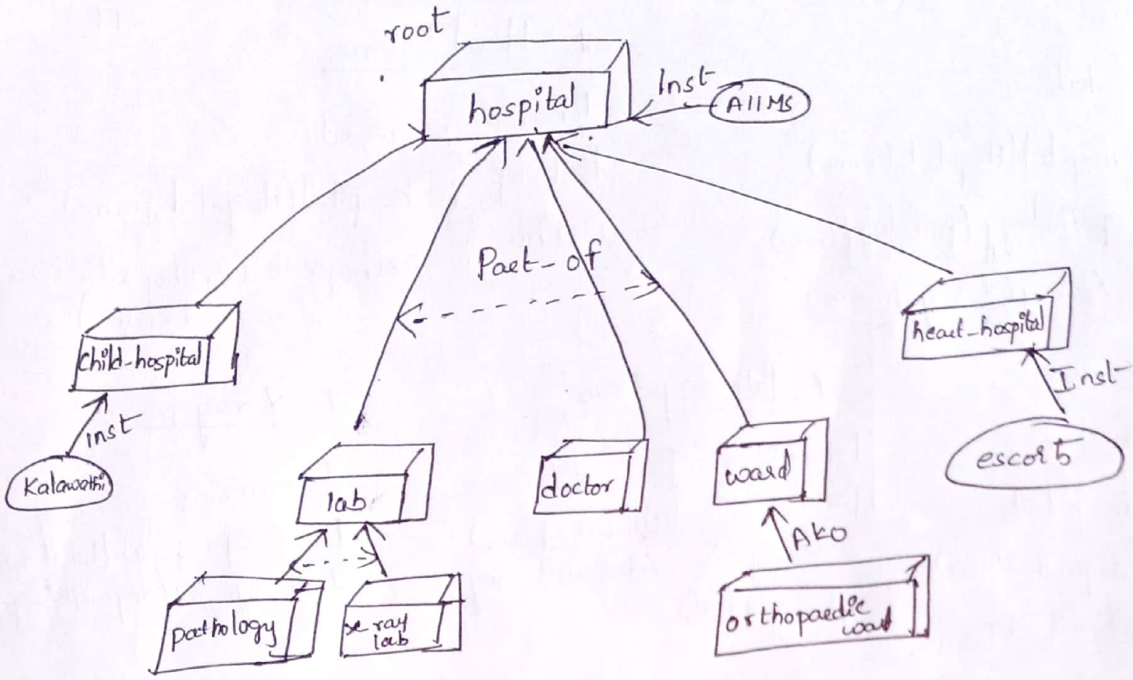
F_name : Kalawathi
Inst : Child_hospital (Child Hospital Frame)
Address : (value-1234 New Delhi)
Phone : (value-01165 73891)
Director : Dr Mike

Heart hospital Frame instance

F_name : escorts
Inst : heart_hospital (Heart Hospital Frame)
Phn_No : (value-0114563732)
Address : (value-Faridabad)

The graphical representation of a frame network for the class 'hospital' described above is as show below.

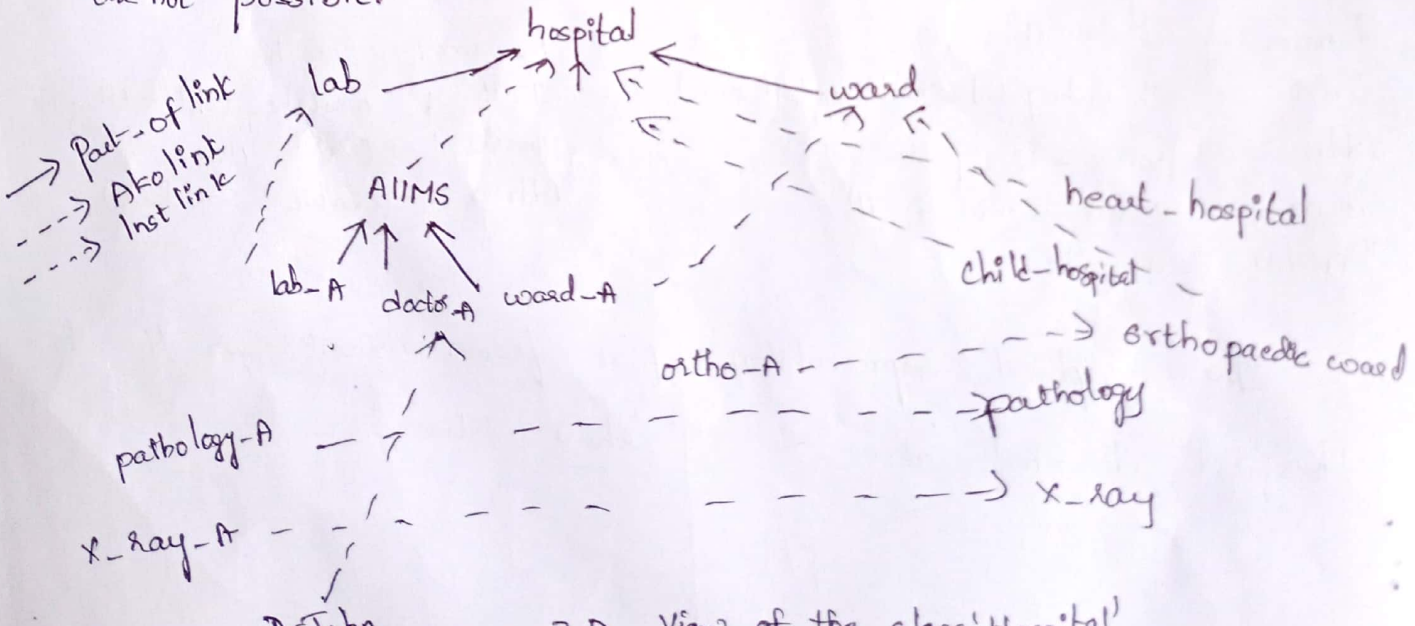
Scanned with CamScanner



A Simple Frame System

A network of frames can be viewed as a 3D figure. one dimensional concerned with the description of a class along with all the classes contained in it & connected Part-of link. The 2D in the hierarchy is of Ako link, while the third Dimension contains the instances of all classes connected with Inst links. It is important to note that each frame can have at most two links with the following combinations:

- A frame can be an instance of some frame & a part of another frame. So it can have both Inst & Part-of links
- A frame can be a-kind-of frame of one frame & a part of another frame. So, it can have Ako & Part-of-links
- However, it is not possible to have a frame which is both an instance and a-kind-of-some class at the same time. So the links Inst & Ako in a frame are not possible.



3-D View of the class 'Hospital'

Scanned with CamScanner

An instance of a hospital is a specific hospital, such as AIIMS, which will have instances of all the classes in the n/w of hospital frame, that is, instances of the lab, doctors & ~~ward~~ ward.

Further, if we wish to represent the 3-D structure of a n/w of frames in first order predicate logic, it could be done as follows:

$$\forall x\,(\text{frame}(x) \equiv \exists y_1, y_2, \ldots y_n((\text{Ako}(x_1,y_1) \vee \text{Inst}(x,y_1) \wedge \text{Part-of}(x,y_2) \wedge \text{slot}_3(x,y_3) \wedge \text{slot}_4(x,y_4) \wedge \ldots \wedge \text{slot}_n(y,y_n))),$$

where, frame(x) means that x is a frame, $\text{slot}_1 \in \{\text{Inst}, \text{Ako}\}$; $\text{slot}_2 = \text{Part-of}$, and $\text{slot}_i(x, y_i)$ means that y_i is value of slot_i of frame x.

- **Inheritance in Frames:**

Inheritance is defined as a mechanism which is utilized for passing knowledge from one frame to other frames down through the taxonomy from general to specific frame. It leads to cognitive economy, where information is only stored in one place, while it can be retrieved from different parts of the n/w.

* **Attachment of Demons (Rules):**

Demons allows us to invoke rules within the frames and are considered to be a powerful style of programming. These can be attached with a slot along with other required facets. For example, an attached demon if-needed may be used for monitoring the behaviour of the system; another demon if-added may be used to validate data when the data is added in value slot.

This allows dynamic information retrieval & storage. One may get values directly from the slot or may have to dynamically calculate the value of the required slot on the basis of other values.

* **Implementation of Frame Knowledge:**

Frames can be easily implemented using OOP languages. Concept of class in OOP can be directly used to code frames that has the concept of a subclass (Ako) & containment (part-of link) of a class within another class. Instances of the frames can be declared as objects of the relevant classes. Slots can be defined as variables & procedural attachment as method. Inheritance is in built in these languages.

- **Representation of Frames in Prolog:** Frames can also be easily implemented using Prolog language. Each frame is represented as a fact in Prolog as frame

frame(F-name, [slot$_1$(facet(value), ...), slot$_2$(facet(value), ...), ...]).

Eg:

Refer pdf.

Scanned with CamScanner

* Inheritance Rules in Prolog:

Since we have coded n/w of frames in Prolog, we need to write inheritance rules as well to obtain relevant information either directly available in frames or through inheritance. Inheritance rules for frame are given below

```
find(X, Y) :-  frame(X, Z), search(Y, Z), !.
find(X,Y) :-   frame(X, [inst(Z) | _]), find(Y,Z), !.
find(X,Y) :-   frame(X, [a|co(Z) | _]), find(Y,Z), !.
find(X,Y) :-   frame(X, [part_of(Z)|_], find(Y,Z).
```

Predicate search will basically retrieve the list of slots_facet pairs & will try to match Y for slots. If a match is found, then its faced value is retrieved, otherwise process is continued till we reach the root frame.

Scanned with CamScanner