INTELLIGENT SYSTEMS (CSE-303-F)

Section B

# Knowledge Representation Techniques

# Knowledge Representation

- *Knowledge representation (KR)* is an important issue in both cognitive science and artificial intelligence.
    - In cognitive science, it is concerned with the way people store and process information and
    - In artificial intelligence (AI), main focus is to store knowledge so that programs can process it and achieve human intelligence.
- There are different ways of representing knowledge e.g.
    - predicate logic,
    - semantic networks,
    - extended semantic net,
    - frames,
    - conceptual dependency etc.
- In predicate logic, knowledge is represented in the form of rules and facts as is done in Prolog.

# Semantic Network

- Formalism for representing information about objects, people, concepts and specific relationship between them.
- The syntax of semantic net is simple. It is a network of labeled nodes and links.
  - It's a directed graph with nodes corresponding to concepts, facts, objects etc. and
  - arcs showing relation or association between two concepts.
- The commonly used links in semantic net are of the following types.
  - **isa** → subclass of entity (e.g., child hospital is subclass of hospital)
  - **inst** → particular instance of a class (e.g., India is an instance of country)
  - **prop** → property link (e.g., property of dog is 'bark)

# Representation of Knowledge in Sem Net

"Every human, animal and bird is living thing who breathe and eat. All birds can fly. All man and woman are humans who have two legs. Cat is an animal and has a fur. All animals have skin and can move. Giraffe is an animal who is tall and has long legs. Parrot is a bird and is green in color".

# Representation in Predicate Logic

- Every human, animal and bird is living thing who breathe and eat.
  $\forall$X [human(X) $\rightarrow$ living(X)]
  $\forall$X [animal(X) $\rightarrow$ living(X)]
  $\forall$X [bird(X) $\rightarrow$ living(X)]
- All birds are animal and can fly.
  $\forall$X [bird(X) $\wedge$ canfly(X)]
- Every man and woman are humans who have two legs.
  $\forall$X [man(X) $\wedge$ haslegs(X)]
  $\forall$X [woman(X) $\wedge$ haslegs(X)]
  $\forall$X [human(X) $\wedge$ has(X, legs)]

- Cat is an animal and has a fur.
  animal(cat) $\wedge$ has(cat, fur)
- All animals have skin and can move.
  $\forall$X [animal(X) $\rightarrow$ has(X, skin) $\wedge$ canmove(X)]
- Giraffe is an animal who is tall and has long legs.
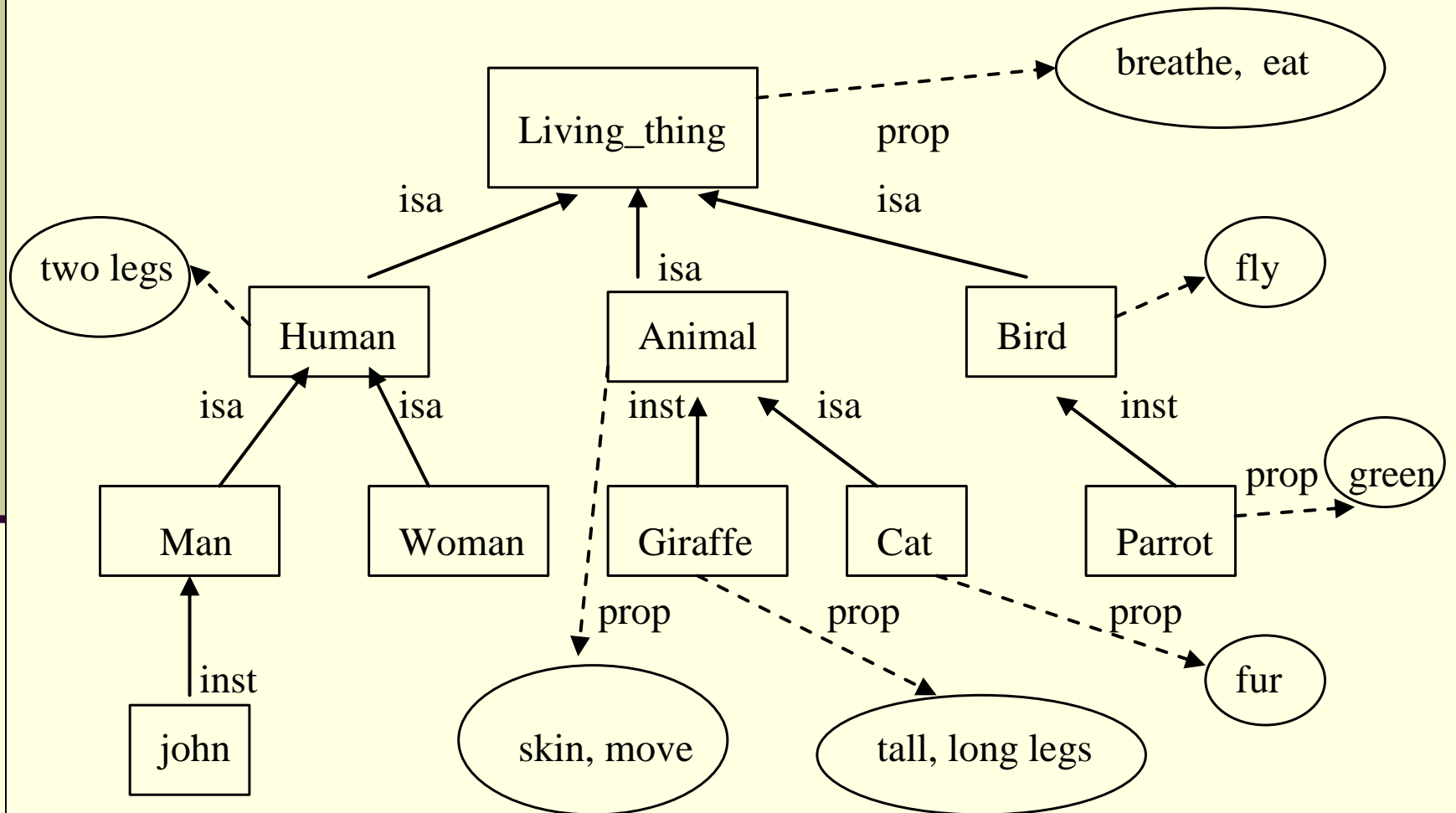  animal(giraffe) $\wedge$ has(giraffe, long_legs) $\wedge$ is(giraffe, tall)
- Parrot is a bird and is green in color.
  bird(parrot) $\wedge$ has(parrot, green_colour)

# Representation in Semantic Net

**Semantic Net**

# Inheritance

- Inheritance mechanism allows knowledge to be stored at the highest possible level of abstraction which reduces the size of knowledge base.
    - It facilitates inferencing of information associated with semantic nets.
    - It is a natural tool for representing taxonomically structured information and ensures that all the members and sub-concepts of a concept share common properties.
    - It also helps us to maintain the consistency of the knowledge base by adding new concepts and members of existing ones.
- Properties attached to a particular object (class) are to be inherited by all subclasses and members of that class.

# Property Inheritance Algorithm

**Input:** Object, and property to be found from Semantic Net;

**Output:** Yes, if the object has the desired property else return false;

**Procedure:**

- Find an object in the semantic net;   Found = false;
- While {(object ≠ root) OR Found } DO
  {  If there is a a property attribute attached with an object then
        {  Found = true;  Report 'Yes'}                else
        object=inst(object, class) OR isa(object, class)
  };
- If Found = False then report 'No';   Stop

# Coding of Semantic Net in Prolog

| Isa facts | Instance facts | Property facts |
|---|---|---|
| isa(living_thing, nil). | inst(john, man). | prop(breathe, living_thing). |
| isa(human, living_thing). | inst(giraffe, animal). | prop(eat, living_thing). |
| isa(animals, living_thing). | inst(parrot, bird) | prop(two_legs, human). |
| isa(birds, living_thing). | | prop(skin, animal). |
| isa(man, human ). | | prop(move, animal). |
| isa(woman, human). | | prop(fur, bird). |
| isa(cat, animal). | | prop(tall, giraffe). |
| | | prop(long_legs, giraffe). |
| | | prop(tall, animal). |
| | | prop(green, parrot). |

# Inheritance Rules in Prolog

**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).

# Queries

- Is john human?
- Is parrot a living thing?
- Is giraffe an aimal?
- Is woman subclassof living thing
- Does parrot fly?
- Does john breathe?
- has parrot fur?
- Does cat fly?

?- instance(john, humans). Y
?- instance (parrot, living_thing).                Y
?- instance (giraffe, animal).Y
?- subclass(woman, living_things).                Y
?- property(fly, parrot).      Y
?- property (john, breathe).  Y
?- property(fur, parrot).      N
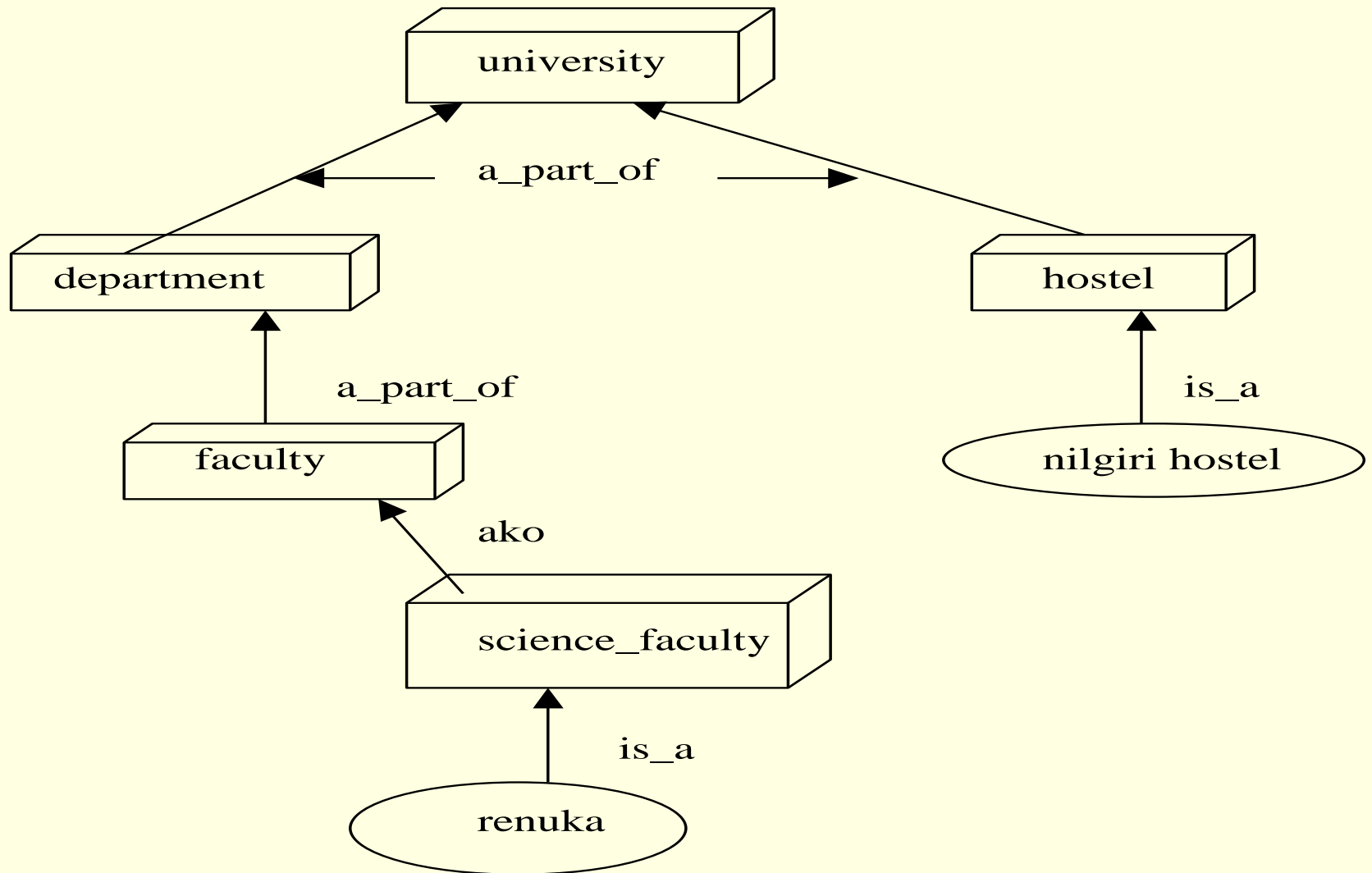?- property(fly, cat).          N

# Knowledge Representation using Frames

- Frames are more structured form of packaging knowledge,
  - used for representing objects, concepts etc.
- Frames are organized into hierarchies or network of frames.
- Lower level frames can inherit information from upper level frames in network.
- Nodes are connected using links viz.,
  - **ako / subc** (links two class frames, one of which is subclass of other e.g., science_faculty class is **ako** of faculty class),
  - **is_a / inst** ( connects a particular instance of a class frame e.g., Renuka **is_a** science_faculty)
  - **a_part_of** (connects two class frames one of which is contained in other e.g., faculty class **is_part_of** department class).
  - Property link of semantic net is replaced by SLOT fields.
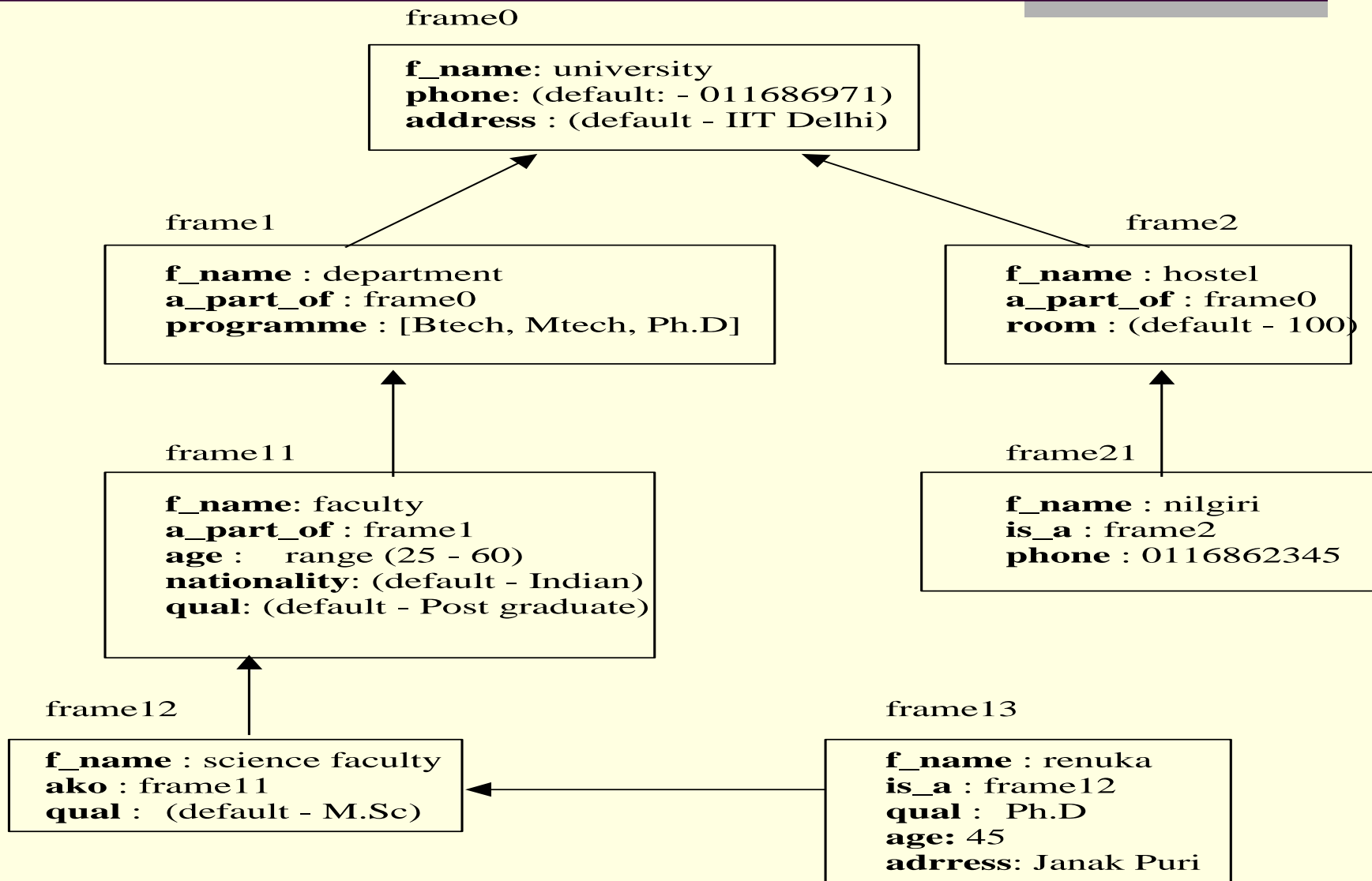
# Cont…

- A frame may have any number of slots needed for describing object. e.g.,
  - faculty frame may have name, age, address, qualification etc as slot names.
- Each frame includes two basic elements : slots and facets.
  - Each slot may contain one or more **facets** (called fillers) which may take many forms such as:
    - **value** (value of the slot),
    - **default** (default value of the slot),
    - **range** (indicates the range of integer or enumerated values, a slot can have),
    - **demons** (procedural attachments such as if_needed, if_deleted, if_added etc.) and
    - **other** (may contain rules, other frames, semantic net or any type of other information).

# Frame Network - Example

# Detailed Representation of Frame Network

frame0

**f_name**: university
**phone**: (default: – 011686971)
**address** : (default – IIT Delhi)

frame1

**f_name** : department
**a_part_of** : frame0
**programme** : [Btech, Mtech, Ph.D]

frame2

**f_name** : hostel
**a_part_of** : frame0
**room** : (default – 100)

frame11

**f_name**: faculty
**a_part_of** : frame1
**age** :    range (25 – 60)
**nationality**: (default – Indian)
**qual**: (default – Post graduate)

frame21

**f_name** : nilgiri
**is_a** : frame2
**phone** : 0116862345

frame12

**f_name** : science faculty
**ako** : frame11
**qual** :  (default – M.Sc)

frame13

**f_name** : renuka
**is_a** : frame12
**qual** :  Ph.D
**age:** 45
**adrress**: Janak Puri

# Description of Frames

- Each frame represents either a class or an instance.
- Class frame represents a general concept whereas instance frame represents a specific occurrence of the class instance.
- Class frame generally have default values which can be redefined at lower levels.
- If class frame has actual value facet then decedent frames can not modify that value.
- Value remains unchanged for subclasses and instances.

# Inheritance in Frames

- Suppose we want to know nationality or phone of an instance-frame frame13 of renuka.

- These informations are not given in this frame.

- Search will start from  frame13 in upward direction  till we get our answer or have reached root frame.

- The frames can be easily represented in prolog by choosing predicate name as frame with two arguments.

- First argument is the name of the frame and second argument is a list of slot - facet pair.

# Coding of frames in Prolog

**frame**(university, [phone (default, 011686971),
address (default, IIT Delhi)]).

**frame**(deaprtment, [a_part_of (university),
programme ([Btech, Mtech, Ph.d]))]).

**frame**(hostel, [a_part_of (university),  room(default, 100)]).

**frame**(faculty, [a_part_of (department), age(range,25,60),
nationality(default, indian), qual(default, postgraduate)]).

**frame**(nilgiri, [is_a (hostel), phone(011686234)]).

**frame**(science_faculty, [ako (faculty),qual(default, M.Sc.)]).

**frame**(renuka, [is_a (science_faculty), qual(Ph.D.),
age(45), address(janakpuri)]).

# Inheritance Program in Prolog

find(X, Y) :- frame(X, Z), search(Z, Y), !.

find(X, Y) :- frame(X, [is_a(Z),_]), find(Z, Y), !.

find(X, Y) :- frame(X, [ako(Z), _]), find(Z, Y), !.

find(X, Y) :- frame(X, [a_part_of(Z), _]), find(Z, Y).

- Predicate **search** will basically retrieve the list of slots-facet pair and will try to match Y for slot.
- If match is found then its facet value is retrieved otherwise process is continued till we reach to root frame

# Extended Semantic Network

- In conventional Sem Net, clausal form of logic can not be expressed.
- Extended Semantic Network (ESNet) combines the advantages of both logic and semantic network.
- In the ESNet, terms are represented by nodes similar to Sem Net.
- Binary predicate symbols in clausal logic are represented by labels on arcs of ESNet.
  - An *atom* of the form "Love(john, mary)" is an arc labeled as 'Love' with its two end nodes representing 'john' and 'mary'.
- *Conclusions* and *conditions* in clausal form are represented by different kinds of arcs.
  - Conditions are drawn with two lines ⇐ and conclusions are drawn with one heavy line ⟵ .

# Examples

- Represent 'grandfather' definition

  Gfather(X, Y) ← Father(X, Z), Parent(Z, Y) in ESNet.
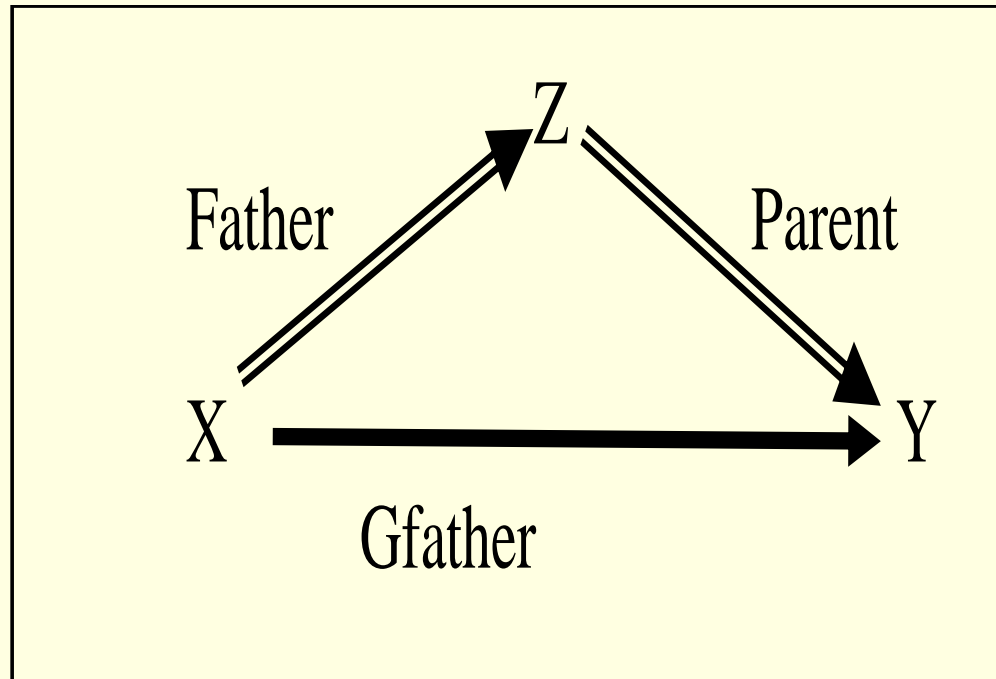
# Cont…Example

- Represent clausal rule **"Male(X), Female(X) ← Human(X)"** using binary representation as "Isa(X, male), Isa(X, female) ← Isa( X, human)" and subsequently in ESNet as follows:
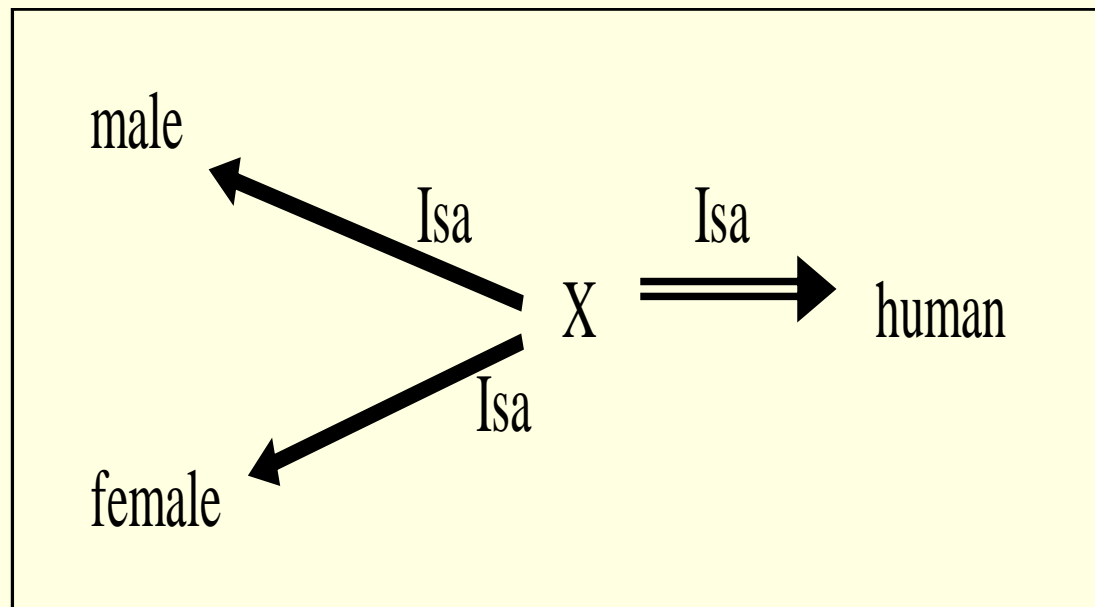
# Inference Rules in ESNet

- Inference rules are embedded in the representation itself.
- The inference that "for every action of giving, there is an action of taking" in clausal logic written as

**"Action(E, take) ← Action(E, give)".**

ESNet

$$E \xrightarrow{\text{Action}} \text{take}$$

$$E \xRightarrow{\text{Action}} \text{give}$$
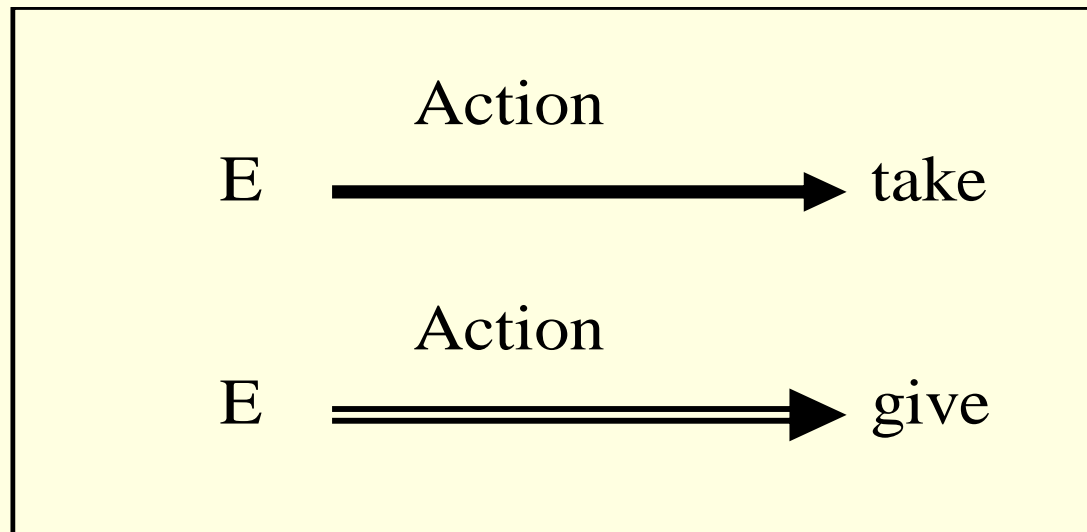
# Cont…

- The inference rule such as "an actor of taking action is also the recipient of the action" can be easily represented in clausal logic as:
  - Here E is a variable representing an event where an action of taking is happening).

**Recipient(E, Y) ← Acton(E, take), Actor (E, Y)**

ESNet
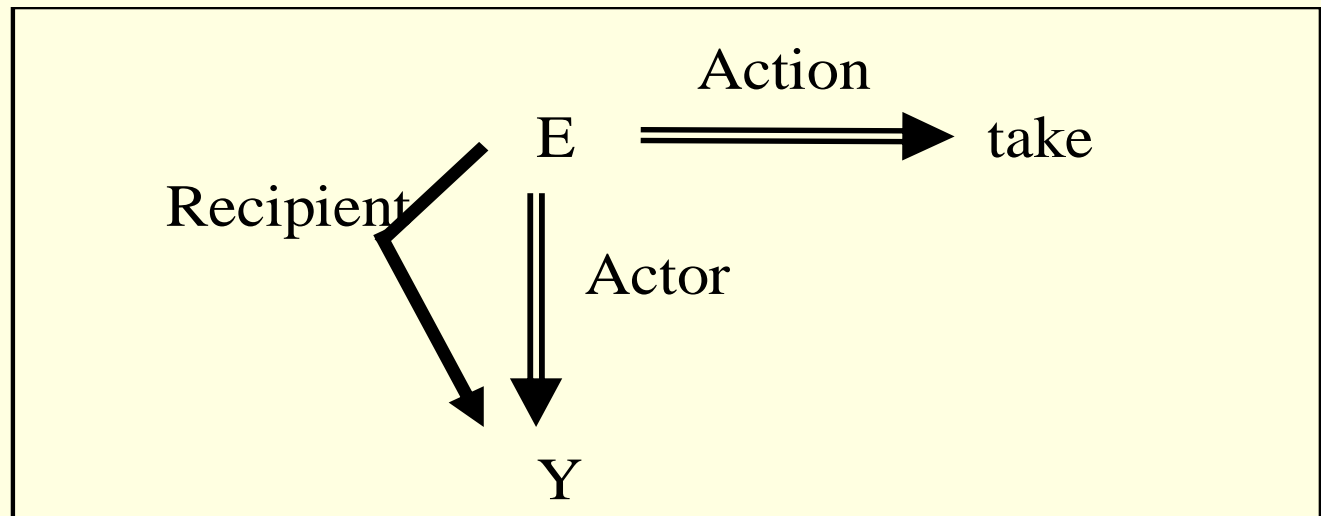
# Example

- Represent the following clauses of Logic in ESNet.

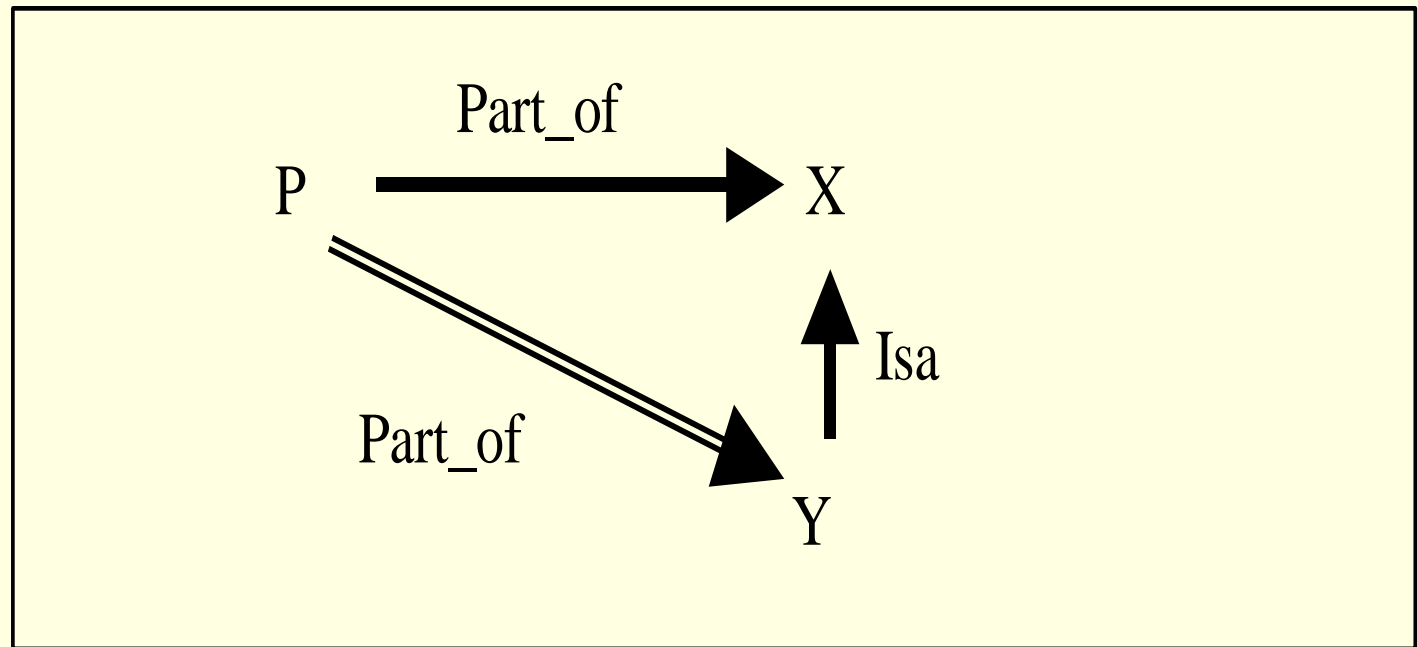  **Recipient(E, Y) ← Acton(E, take), Actor (E, Y)**
  **Object (e, apple).**
  **Action(e, take).**
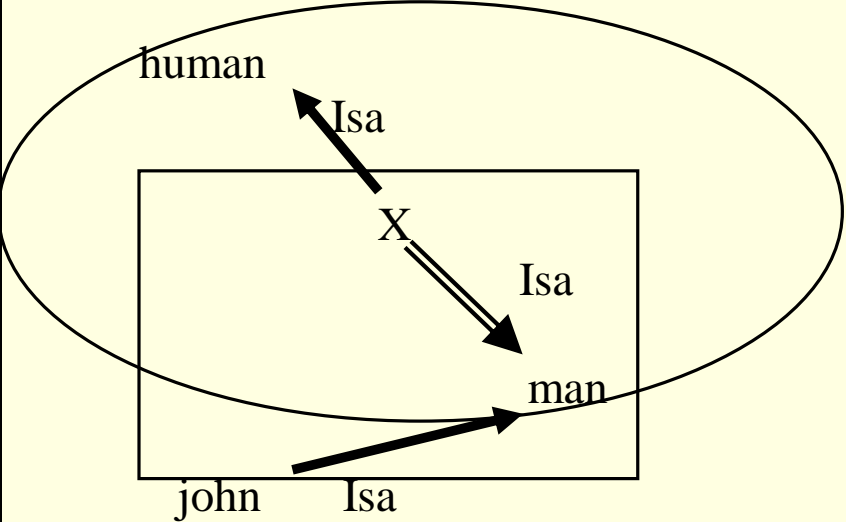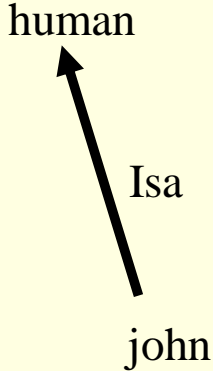  **Actor (e, john) .**
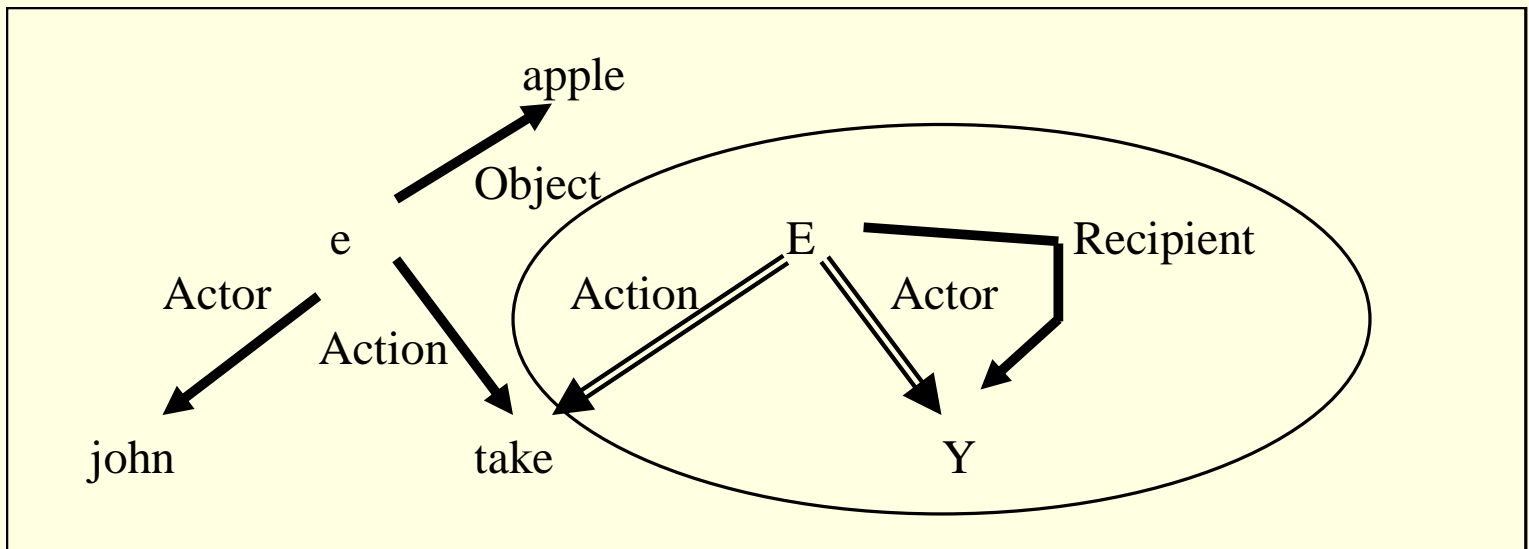
# Contradiction

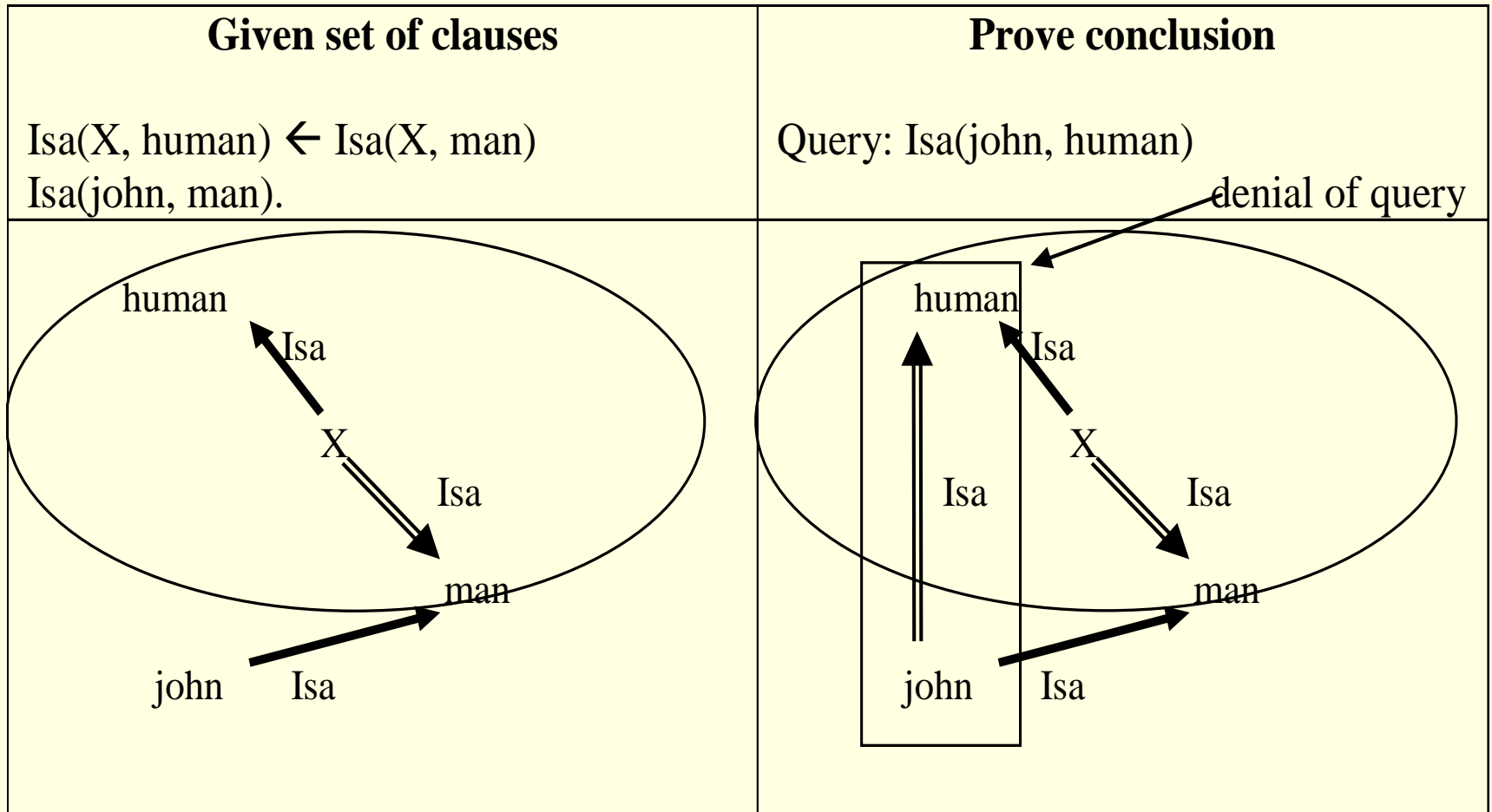- The contradiction in the ESNet arises if we have the following situation.

# Deduction in ESNet

- Both of the following inference mechanisms are available in ESNet.
  - Forward reasoning inference (uses bottom up approach)
    - *Bottom Up Inferencing:* Given an ESNet, apply the following reduction (resolution) using modus ponen rule of logic ({A ← B, B} then A).
  - Backward reasoning inference  (uses top down approach).
    - *Top Down Inferencing:* Prove a conclusion from a given ESNet by adding the denial of the conclusion to the network and show that the resulting set of clauses in the network is inconsistent.
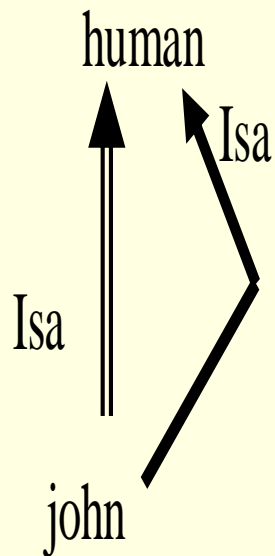
# Example: Bottom Up Inferencing

| Given set of clauses | Inferencing |
|---|---|
| Isa(X, human) ← Isa(X, man)<br>Isa(john, man). | Isa(john, human) |



Here    X is bound to john

# Example: Top Down Inferencing



| Given set of clauses | Prove conclusion |
|---|---|
| Isa(X, human) ← Isa(X, man)<br>Isa(john, man). | Query: Isa(john, human)<br>denial of query |

# Cont…

| | |
|---|---|
| human      X = john<br><br>Isa<br><br>Isa<br><br>john | Contradiction or Empty network is generated. Hence "Isa(john, human)" is proved. |