

Artificial Intelligence

V Sem. CSE

Unit-IV


Knowledge Representation

Course Outcomes

After successful completion of this course, the student will be able to:

CO	Course Outcomes	Knowledge Level
1	Illustrate the concept of Intelligent systems and current trends in AI	K2
2	Apply Problem solving, Problem reduction and Game Playing techniques in AI	K3
3	Illustrate the Logic concepts in AI	K2
4	Explain the Knowledge Representation techniques in AI	K2
5	Describe Expert Systems and their applications	K2
6	Illustrate Uncertainty Measures	K2


Introduction

- Knowledge Representation(KR) is an important issue in both cognitive science and AI.
 - In cognitive science, it is concerned with the way in which information is stored and processed by humans, while in AI, the main focus is on storing knowledge or information in such a manner that programs can process it and achieve human intelligence.
 - Representing knowledge in some specific ways makes certain problems easier to solve.
 - The fundamental goal of knowledge representation is to represent knowledge in a manner that facilitates the process of inferencing(drawing conclusions) from it.
 - Several programming languages oriented to KR have been developed till date
- 

Introduction


- In Prolog, knowledge is represented in the form of rules and facts
- The languages such as SGML, XML, RDF have been developed to represent the structure of documents more explicitly. These languages facilitate processes such as information retrieval, data mining etc.

Any knowledge representation system should possess properties such as :

- 1) **Learning** refers to a capability that acquires new knowledge, behaviours, understanding etc.
 - 2) **Efficiency in acquisition** refers to the ability to acquire new knowledge using automatic methods wherever possible rather than relying on human intervention
 - 3) **Representational adequacy** refers to the ability to represent the required knowledge
 - 4) **Inferential adequacy** refers to the ability of manipulating knowledge to produce new knowledge from the existing one
- 

Introduction

Knowledge representation is a core component of a number of applications such as

- Expert systems
 - Machine translation systems
 - Computer-aided maintenance systems
 - Information retrieval systems
 - Database front-ends
- 

Approaches to Knowledge Representation

- AI programs use structures known as knowledge structures to represent objects, facts, relationships and procedures
- The main function of these knowledge structures is to provide expertise and information so that a program can operate in an intelligent way
- Knowledge structures are usually composed of both traditional and complex structures such as:
 - ✓ Semantic networks
 - ✓ Frames
 - ✓ Scripts
 - ✓ Conceptual dependency structures and so on
- A knowledge base is a special type of database that holds knowledge of the domain. It shares some of the functions of the database systems such as storing, updating and retrieving information; however, they are much powerful and more structured.
- Knowledge bases are organized in hierarchical structures with in-built inheritance and inferencing capabilities

Approaches to Knowledge Representation

1. Relational knowledge

- It comprises objects consisting of attributes and its associated values. The simplest way of storing facts is to use a relational method. For example a fact:
- “John is a male whose age is 38 years and who is a graduate with a salary of Rs.20000” can be easily represented like this:

Name	Age	Gender	Qualification	Salary
John	38	Male	Graduate	20000
Mike	25	Male	Undergraduate	15000
Mary	30	Female	Ph.D.	25000
James	29	Male	Graduate	18000

- This representation helps in storing the facts but gives little opportunity for inferencing.
- We can easily obtain the answers for queries like “*What is the age of John?*” but difficult to answer queries like “*Does a person having a Ph.D qualification earn more?*”
- Inferencing new knowledge is not possible from such structures

Approaches to Knowledge Representation

2. Knowledge Represented as Logic

- Inferential capability can be achieved if knowledge is represented in the form of formal logic.
- For eg. “All humans are mortal” cannot be represented using relational approach; instead it can be easily represented in predicate logic as follows:


$$(\forall X) \text{human}(X) \rightarrow \text{mortal}(X).$$

If we have a fact “John is human” , then we can easily infer that “John is mortal”

- Advantages of this approach are that we can represent a set of rules , derive more facts, truths and verify the correctness of new statements

Approaches to Knowledge Representation

3. Procedural knowledge

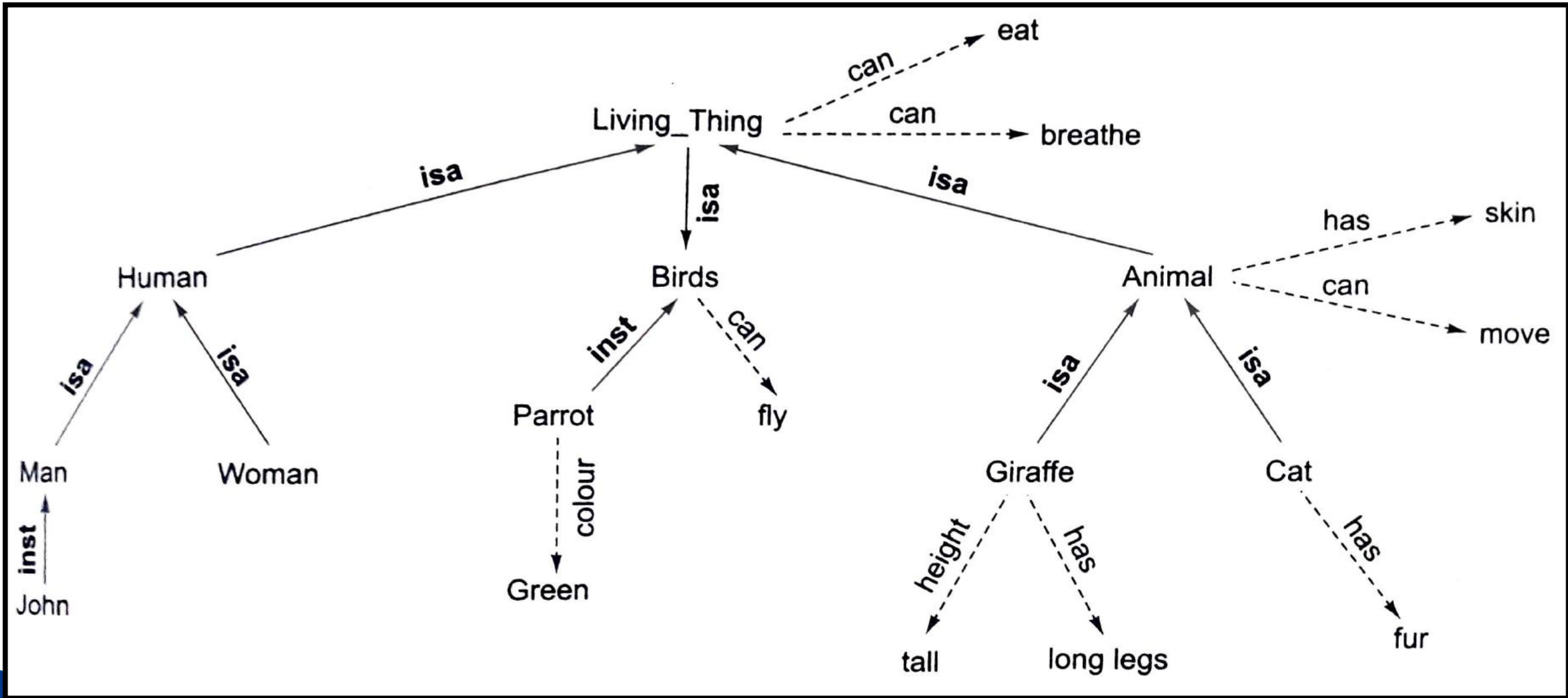
- It is encoded in the form of procedures which carry out specific tasks based on relevant knowledge.
 - For example: An interpreter for a programming language interprets a program on the basis of the available knowledge regarding the syntax and semantics of the language.
 - Advantages of this approach are that domain-specific knowledge can be easily represented and side effects of actions may also be modelled
 - There is a problem of completeness i.e. all cases may not be represented
 - Another problem is consistency i.e. all deductions may not be correct
- 

Knowledge representation using Semantic Network

- We can represent knowledge using a structure called Semantic Network
- Nodes represent concepts or objects and arcs represent relation between two concepts
- It can be represented as a directed graph
- The meaning of a concept is derived from its relationship with other concepts
- For eg.

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

Knowledge representation using Semantic Network

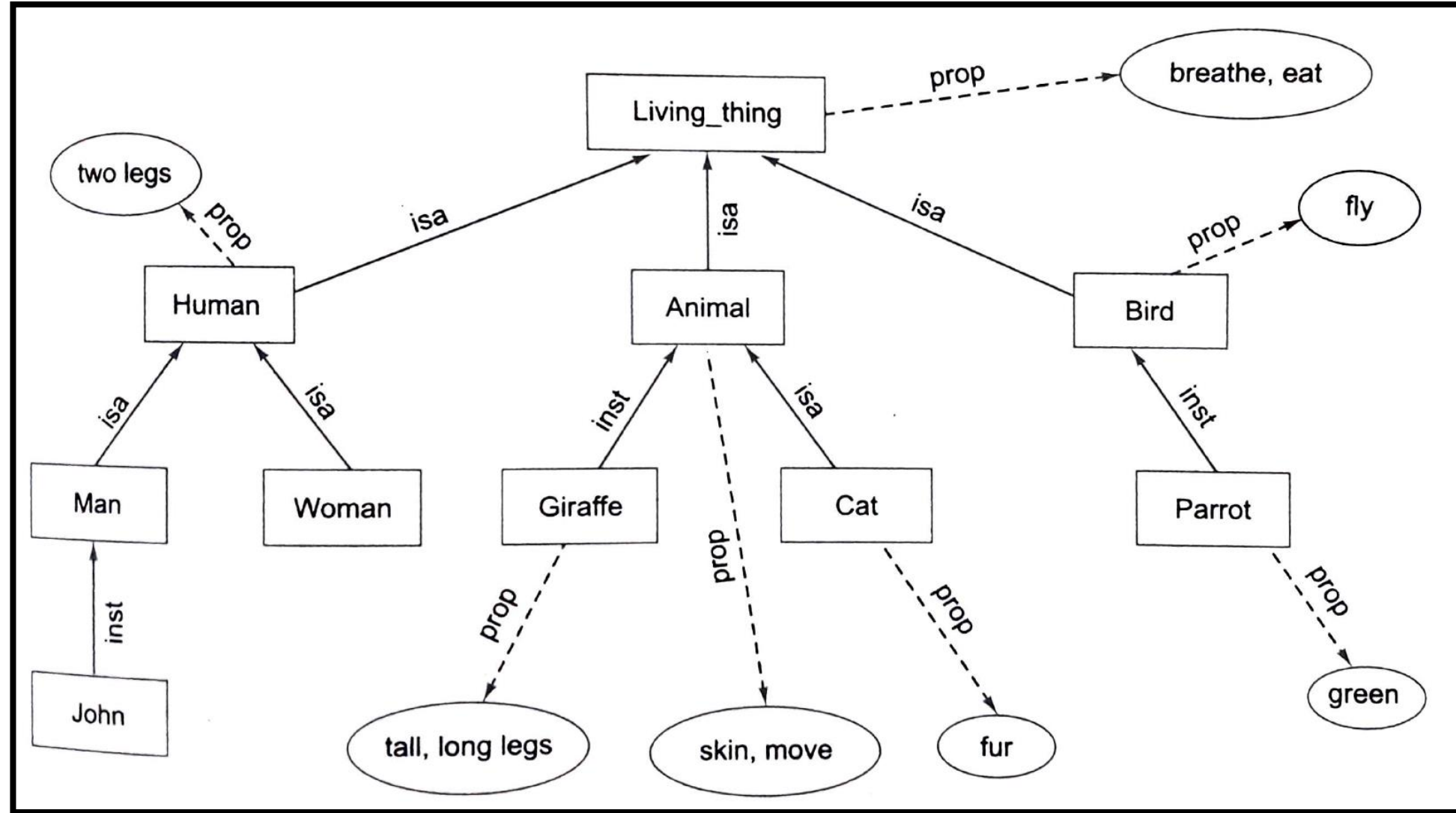


Knowledge representation using Semantic Net

Knowledge representation using Semantic Network


- Two types of relations in the figure are:
 - **isa** : connects two classes, where one concept is a kind or subclass of the other concept
 - **inst**: relates specific members of a class
- Other relations such as {can, has, colour, height} are known as property relations. These have been represented by dotted lines pointing from the concept to its property.
- In this structure, property inheritance is easily achieved.
- For example, the query “Does a parrot breathe?” can be easily answered “yes” even though this property is not directly associated with a parrot.
- The information encoded can be represented as a directed graph as shown below.

Knowledge representation using Semantic Network



Concepts connected with prop links

Inheritance in Semantic Net

- As Semantic net is stored as hierarchical structure, the inheritance mechanism is in-built and facilitates inferencing of information associated with nodes in Semantic Net
 - 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
 - Properties attached to a particular concept can be easily inherited by all sub-concepts and their members
 - 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 the form of facts(relation as predicate and nodes as arguments) and inheritance rules
- 

Inheritance in Semantic Net

Property Inheritance Algorithm

Input: Object and 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 the semantic net;
- Found=False;
- While [(object \neq root) OR ! Found] Do
 - {
 - If there is an attribute attached with an object then Found= true;
 - else { object = isa(object, class) or object= inst(object,class) }}
- If Found = true then report 'Yes' else Report 'No';

Inheritance in Semantic Net

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)

Prolog Facts

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
 - For example, if man is a member of subclass human, then man is also member of living_thing
 - Similarly, an instance of subclass is also an instance of all super classes connected via isa link
 - Further, if John is an instance of man, then we can infer that John is human and also living_thing. Similarly, a property of a class can be inherited by lower sub-classes
 - Prolog rules for handling inheritance in Semantic net is as follows:
- 

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)

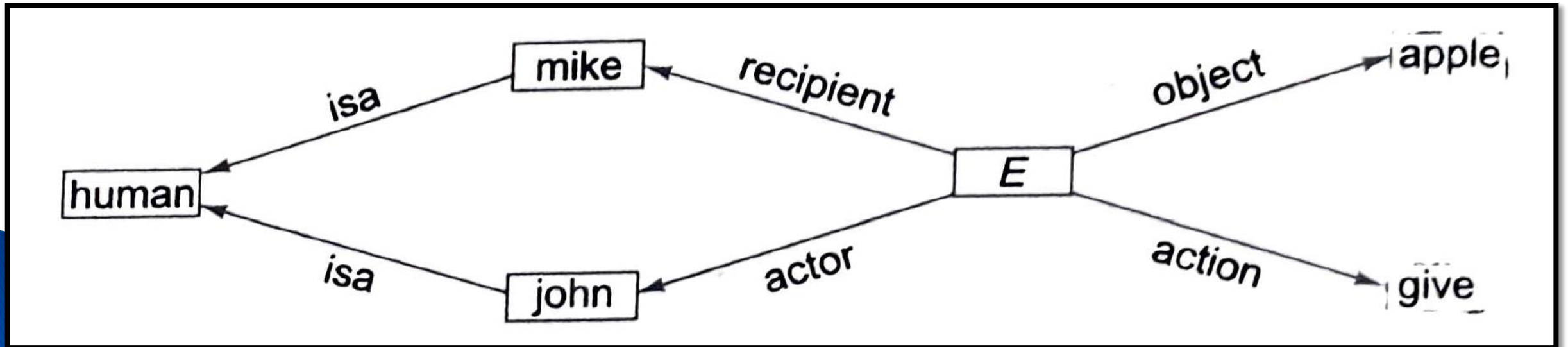
property(X,Y) :- subclass(Y,Z), property(X,Z)

Various Queries for Inheritance program


English query	Prolog goal	Output
Is John human?	?-instance(john, humans)	Yes
Is parrot a living thing?	?-instance(parrot, living_thing)	Yes
Is giraffe an animal?	?-instance(giraffe, animal)	Yes
Is woman a subclass of living thing?	?-subclass(woman, 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 and Semantic networks are two different formalisms that can be used for Knowledge representations
- Simple Semantic network is represented as directed graph whose nodes represent concepts or objects and arcs represent relationships between concepts or objects
- It can only express collections of variable-free assertions.
- The English sentences “John gives an apple to Mike and John and Mike are human may be represented in semantic networks as below:



Extended Semantic Networks for KR

- Here E represents an event which is an act of giving, whose actor is John, object is an apple, and recipient is Mike
 - The relationships in the network can be expressed in clausal form of logic as follows:
 - object(E, apple)
 - action(E, give)
 - actor(E, john)
 - recipient(E, mike)
 - isa(John, human)
 - isa(Mike, human)
- 

Extended Semantic Networks for KR

- Predicate relations corresponding to semantic networks always have two arguments. Therefore, the entire semantic net can be coded using binary representation
- Such representation is advantageous when additional information is added to the given system, for eg. 'John gives an apple to Mike in the kitchen', it is easy to add **location(E, kitchen)**
- In first-order predicate logic, predicate relation can have n arguments, where $n \geq 1$
- For eg. the sentence "John gives an apple to Mike" is easily represented in predicate logic by `give(John, Mike, apple)`. Here John, Mike and apple are arguments, while **give** represents a predicate relation


Extended Semantic Networks for KR

- The predicate logic representation has greater advantages compared to semantic net representation as it can express general propositions, in addition to simple assertions.
- For example, the sentence “John gives an apple to everyone he likes” expressed in predicate logic by the clause:


give(john, X, apple) \leftarrow likes(john, X).

Here the symbol X is a variable representing any individual

Extended Semantic Networks for KR

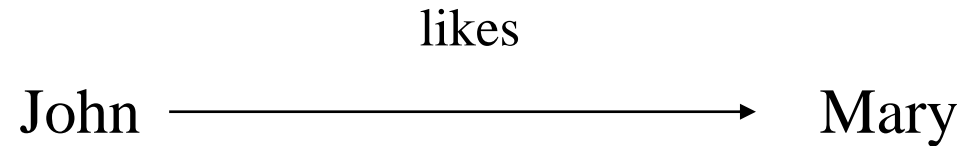
- Despite all the advantages, it is not convenient to add new information in an n-ary representation of Predicate Logic
 - Further, a clause in logic can have several conditions, all of which must hold for the conclusion to be true and can have several alternative conclusions, at least one of which must hold if all the conditions are true
 - In conventional semantic network, we cannot express clausal form of logic. To overcome this shortcoming, R Kowalski and his colleagues proposed an Extended Semantic Network(ESNet) that combines the advantages of both logic and semantic networks
- 

Extended Semantic Networks for KR

- ESNet can be interpreted as variant syntax for the clausal form of logic
 - It has the same expressive power as that of predicate logic with well defined semantics, inference rules and a procedural interpretation
 - It also incorporates the advantages of using **binary relation** as in Semantic Network rather than n-ary relations of logic
 - ESNet is a much **powerful representation** as compared to logic and semantic network
 - In ESNet, the terms are represented by nodes similar to as done in conventional semantic network; constant, variable and functional terms are represented by **constant, variable and functional nodes**, respectively
- 

Extended Semantic Networks for KR

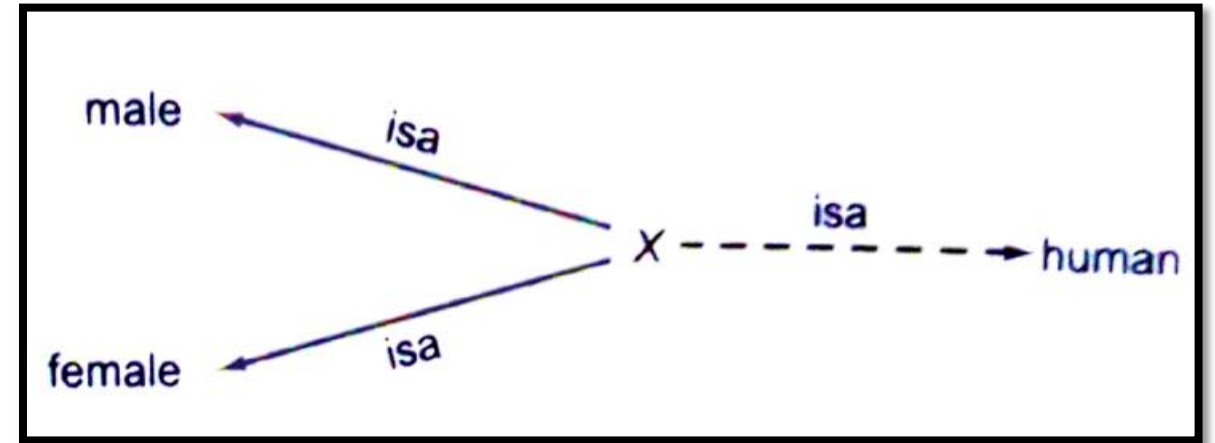
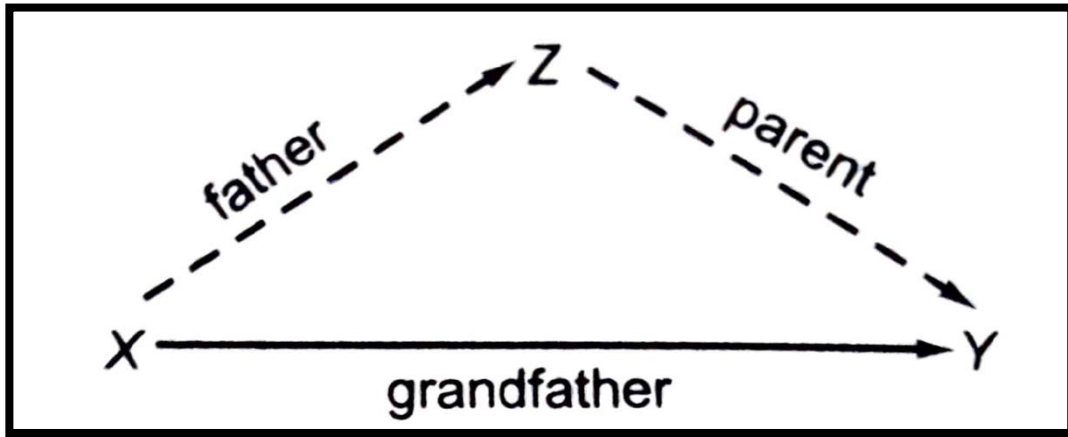
- Binary predicate symbols in clausal logic are represented by labels on arcs of ESNet
- An atom of the form likes(John, Mary) can be represented as below:



- Conclusions and 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. These are called denial links($- - - - - \blacktriangleright$) combines the advantages of both logic and semantic networks.
- The arcs denoting conclusions(positive atom) are drawn with continuous arrow lines. These are known as assertion links(\longrightarrow)

Extended Semantic Networks for KR

For example the rule: **grandfather(X,Y) \leftarrow father(X,Z), parent(Z,Y)** can be represented in ESNet as given below:



Similarly, the clausal rule : **male(X), female(X) \leftarrow human(X)** can be represented using binary representation as **isa(X, male), isa(X, female) \leftarrow isa(X, human)** as above

Inference Rules : ESNet

- The representation of the inference for every action of **giving**, there is an action of **taking** in clausal logic is:

$$\text{action}(f(X), \text{take}) \leftarrow \text{action}(X, \text{give}).$$

- The interpretation of this rule is that the event of **taking** action is a function of the event of giving action. It can be represented in ESNet as below:

$$\begin{array}{ccc} & \text{action} & \\ f(X) & \longrightarrow & \text{take} \end{array}$$

$$\begin{array}{ccc} & \text{action} & \\ X & \text{-----} \blacktriangleright & \text{give} \end{array}$$

ESNet Representation

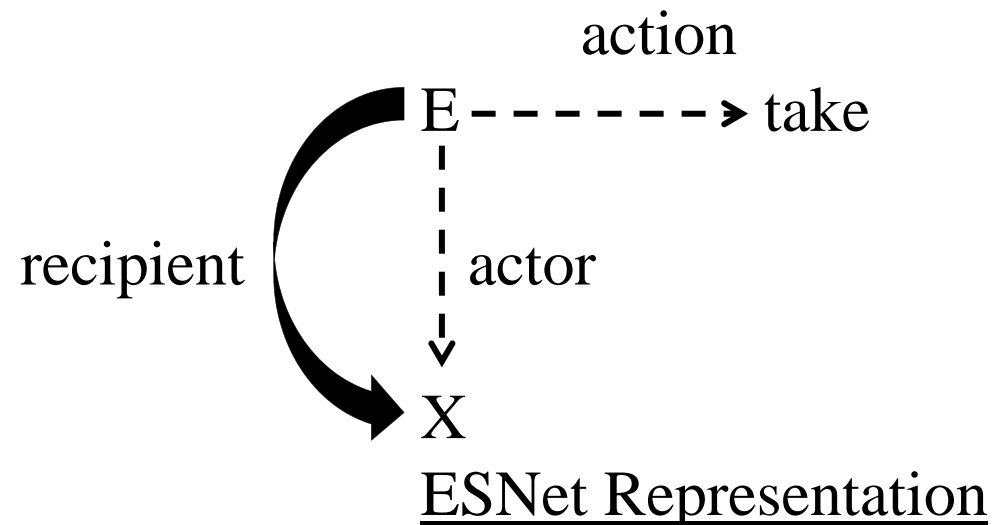
Inference Rules : ESNet

- The inference rule that an actor who performs a taking action is also the recipient of this action and can be represented in clausal logic as:

$$\text{recipient}(E,X) \leftarrow \text{action}(E, \text{take}), \text{actor}(E,X)$$

Here E is a variable representing an event of an action of taking

- It can be represented in ESNet as below:



Inference Rules : ESNet

Example: Represent the following clauses in ESNet:


recipient(E,X) \leftarrow action(E, take), actor(E,X)

object(e , apple)

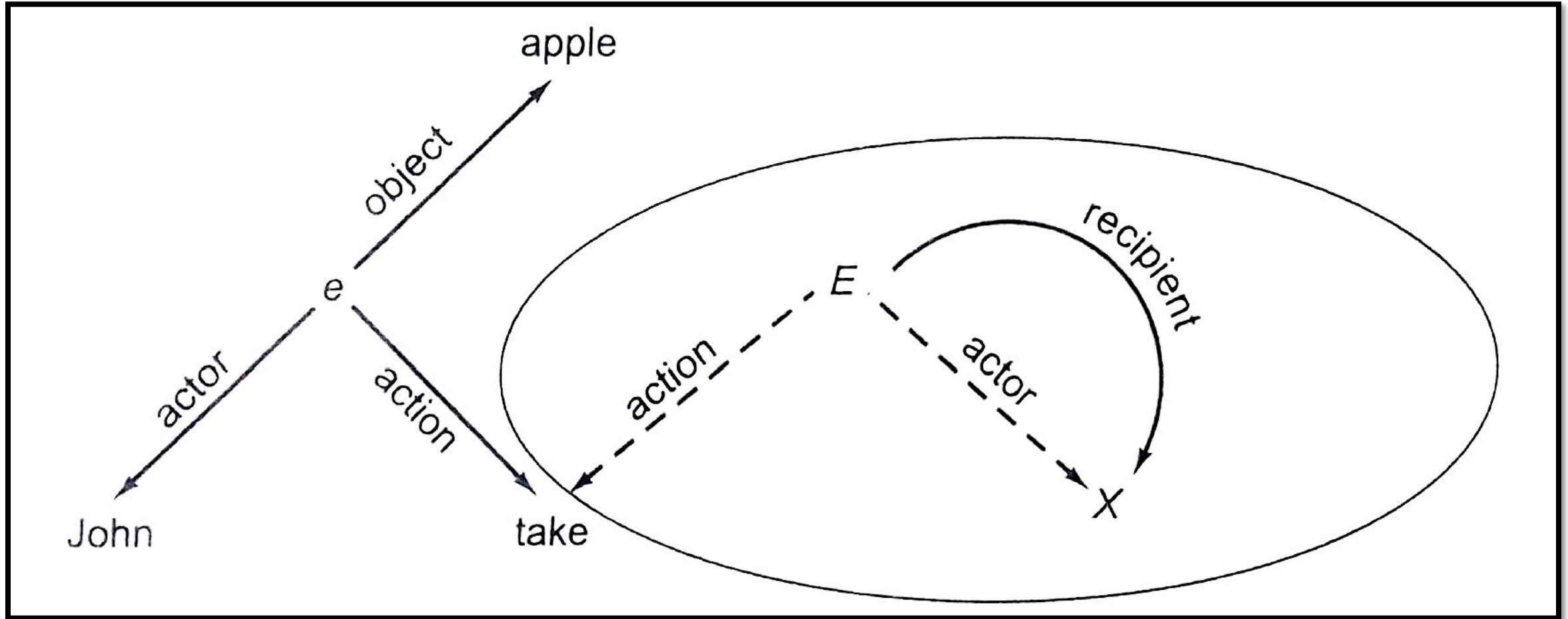
action(e, take)

actor(e, john)

Solution: Here, E is a variable for some event and 'e' is an actual event. ESNet representation is shown below. Here, the partition is shown to delimit the clause pictorially by enclosing the clausal rule in an ellipse.



Inference Rules : ESNet



ESNet Representation

Inference Rules : ESN_{et}

- The hierarchy links , isa, inst, and part_of (or prop) are available in Semantic Networks and are also available as special cases in ESN_{et}
- For eg. Every human has two legs could be represented using

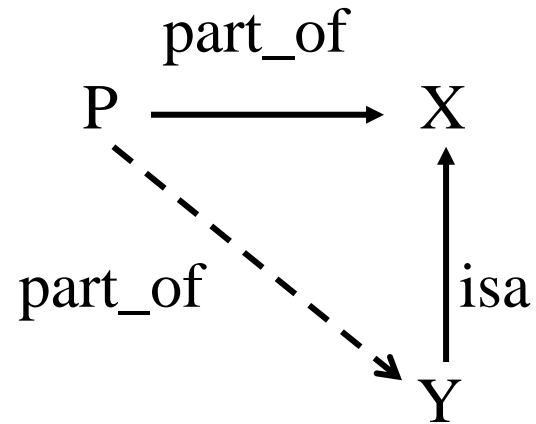
part_of
two_legs \longrightarrow Human

The interpretation is that for every human, there exists two legs which are part_of that human

Inference Rules : ESNet

Contradiction

- The Contradiction in ESNet can be represented as:




Contradiction in ESNet

- Here 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 and hence there is a contradiction in ESNet

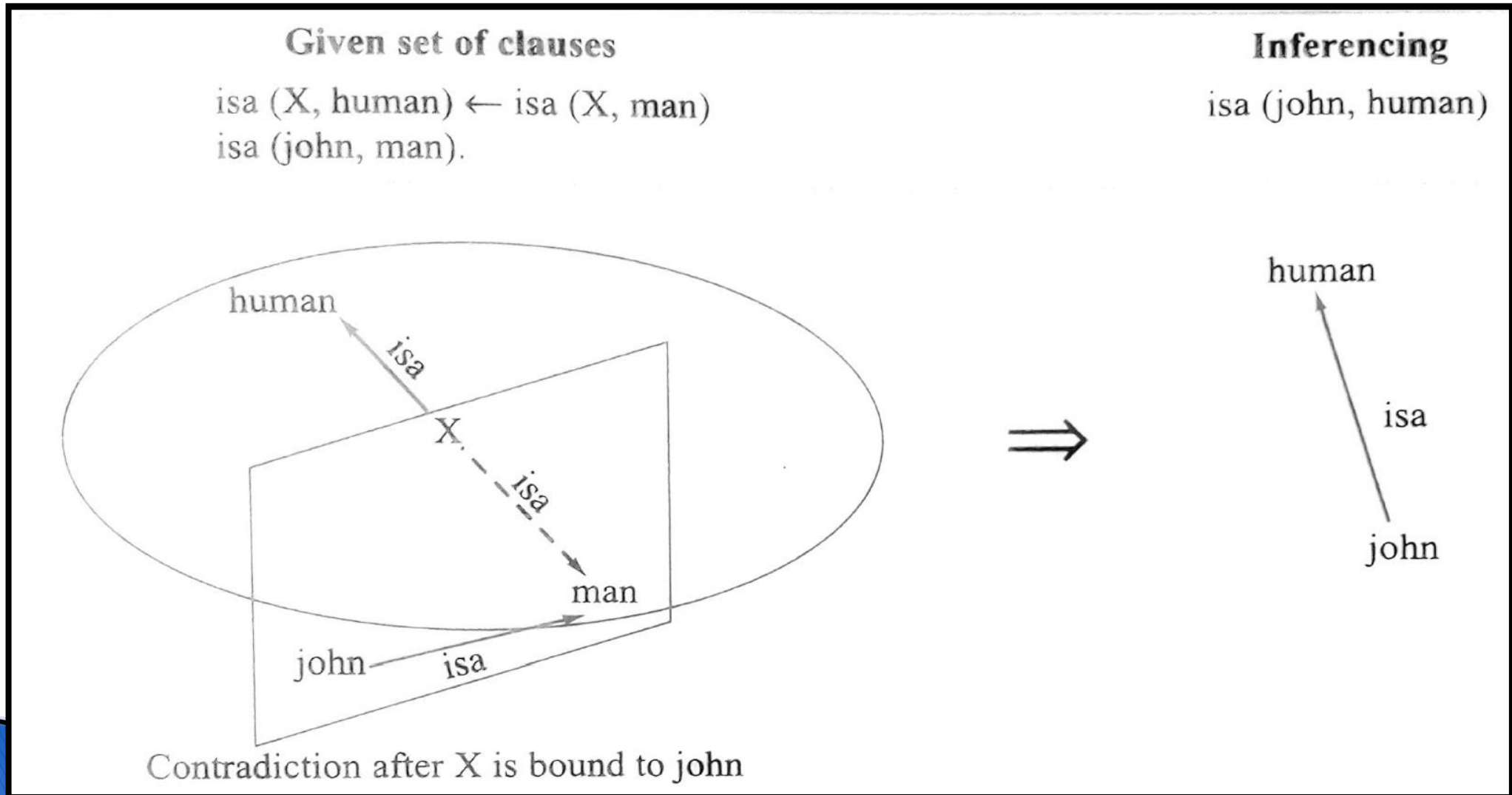
Deduction in Extended Semantic Networks

There are two types of inference mechanisms in ESNet, namely

- **Forward reasoning inference mechanism:** Also called bottom-up approach, in clausal logic derives new assertions from old ones, i.e. we start with given assertions and derive new assertions using clausal rule
 - **Backward reasoning inference mechanism:** Also called top-down approach, we prove the query from the set of clauses using resolution refutation method in clausal logic
- 

Forward Reasoning Inference

Given an ESNet, apply the following reduction (resolution) using Modus Ponen Rule of logic :

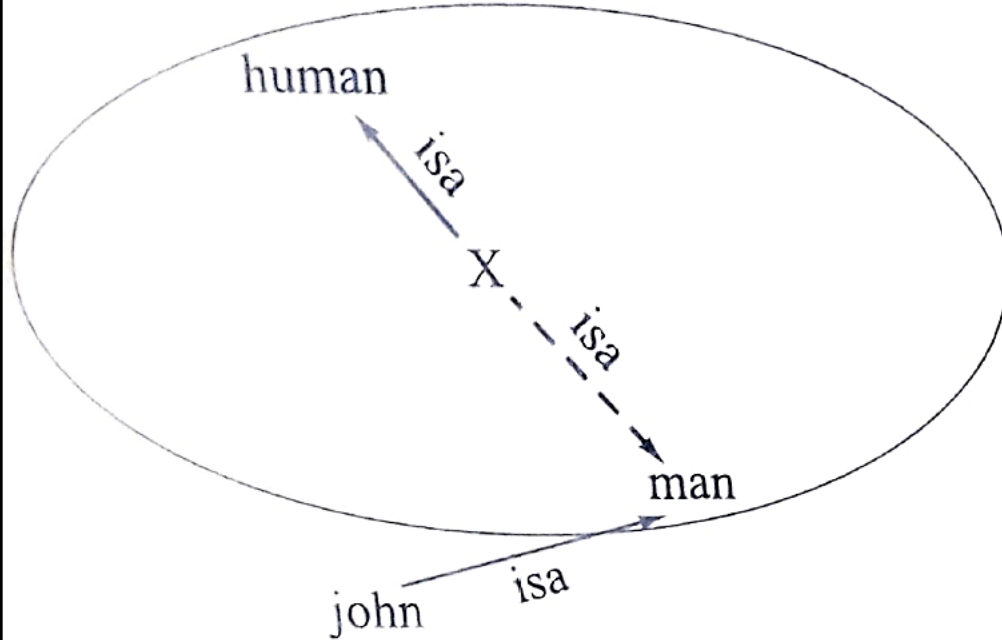


Backward Reasoning Inference

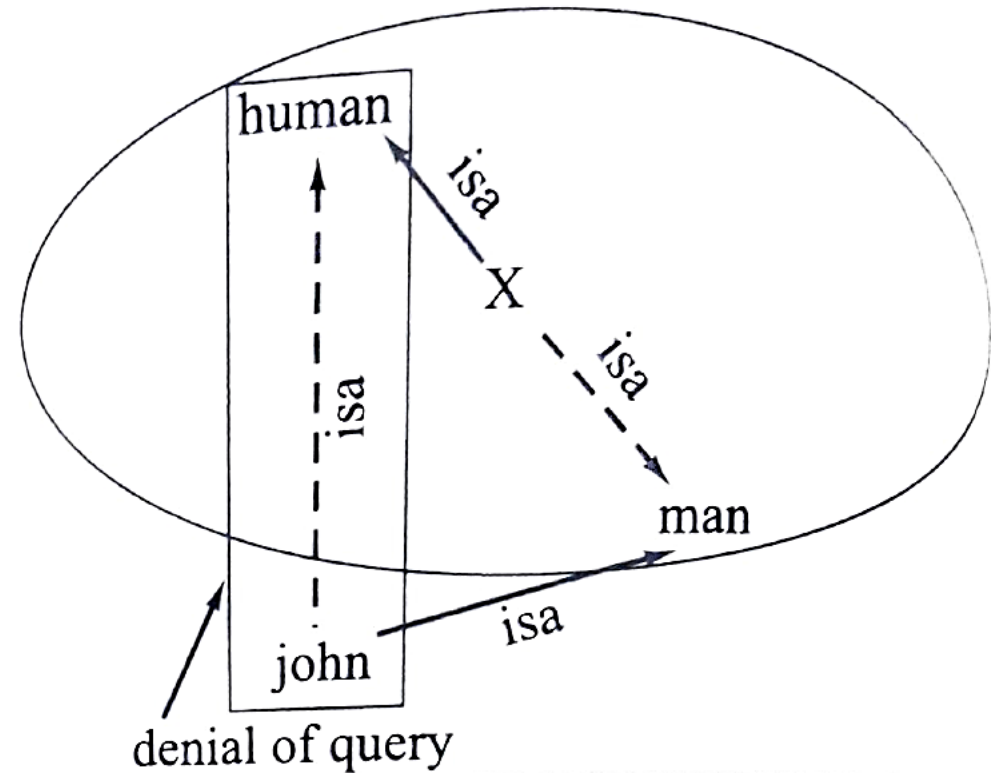
- In this mechanism, we can prove a conclusion or goal from a given ESNet by adding the denial of the conclusion to the network and show that the resulting set of clauses in the network gives contradiction. This is done by performing successive steps of resolution until an explicit contradiction is generated.

Backward Reasoning Inference

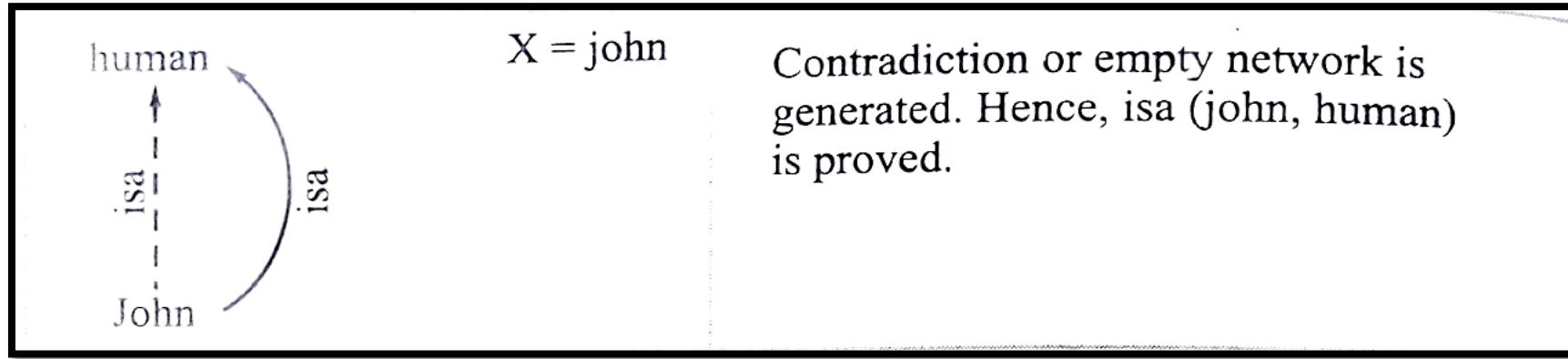
Given set of clauses
 $\text{isa}(X, \text{human}) \leftarrow \text{isa}(X, \text{man})$
 $\text{isa}(\text{john}, \text{man})$



Prove conclusion
Query : $\text{isa}(\text{john}, \text{human})$



Backward Reasoning Inference



Reduction of ESNet

Examples of Inferencing methods

Consider a example represented in clausal form:

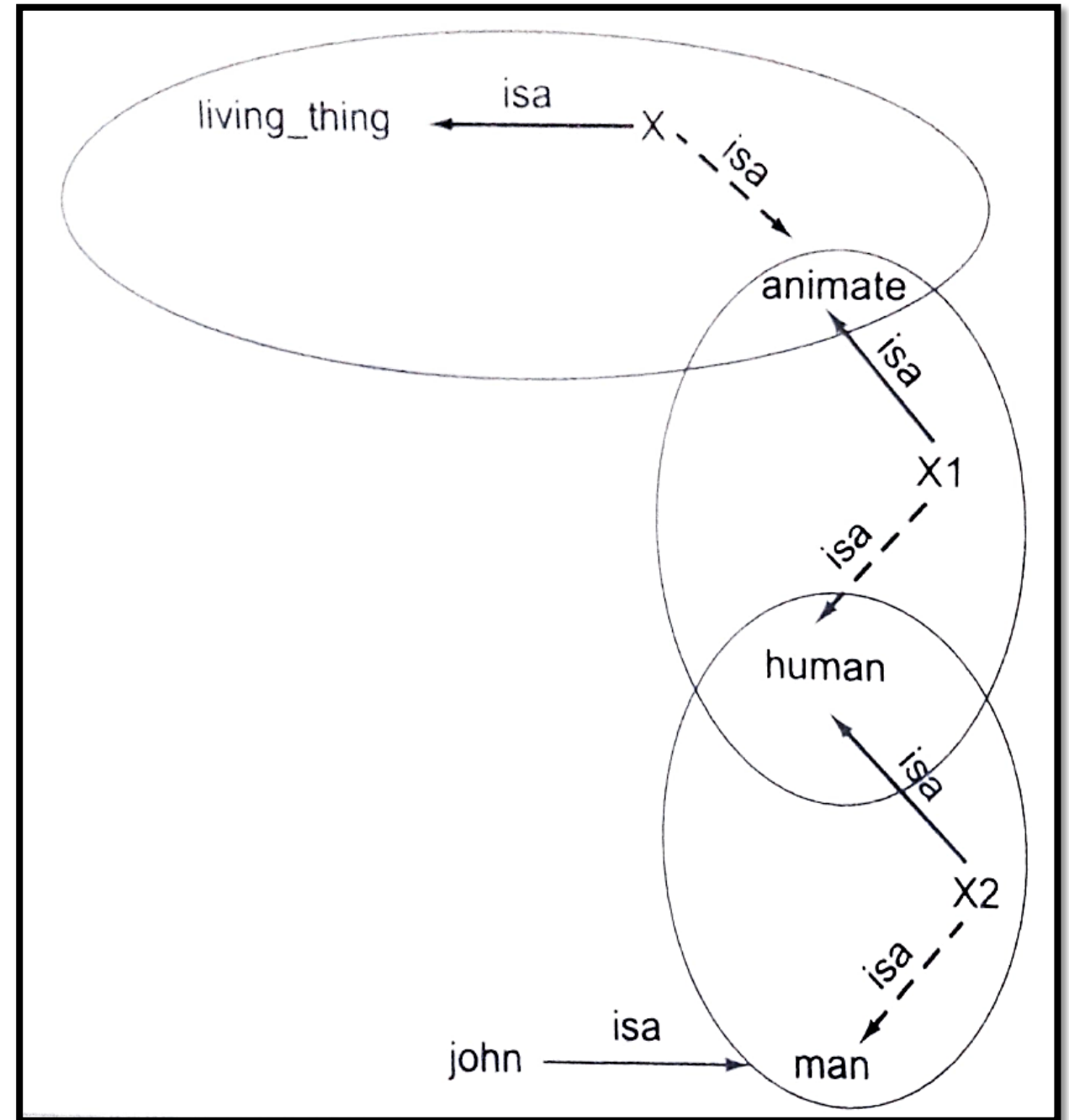
$\text{isa}(X, \text{living_thing}) \leftarrow \text{isa}(X, \text{animate})$

$\text{isa}(X, \text{animate}) \leftarrow \text{isa}(X, \text{human})$

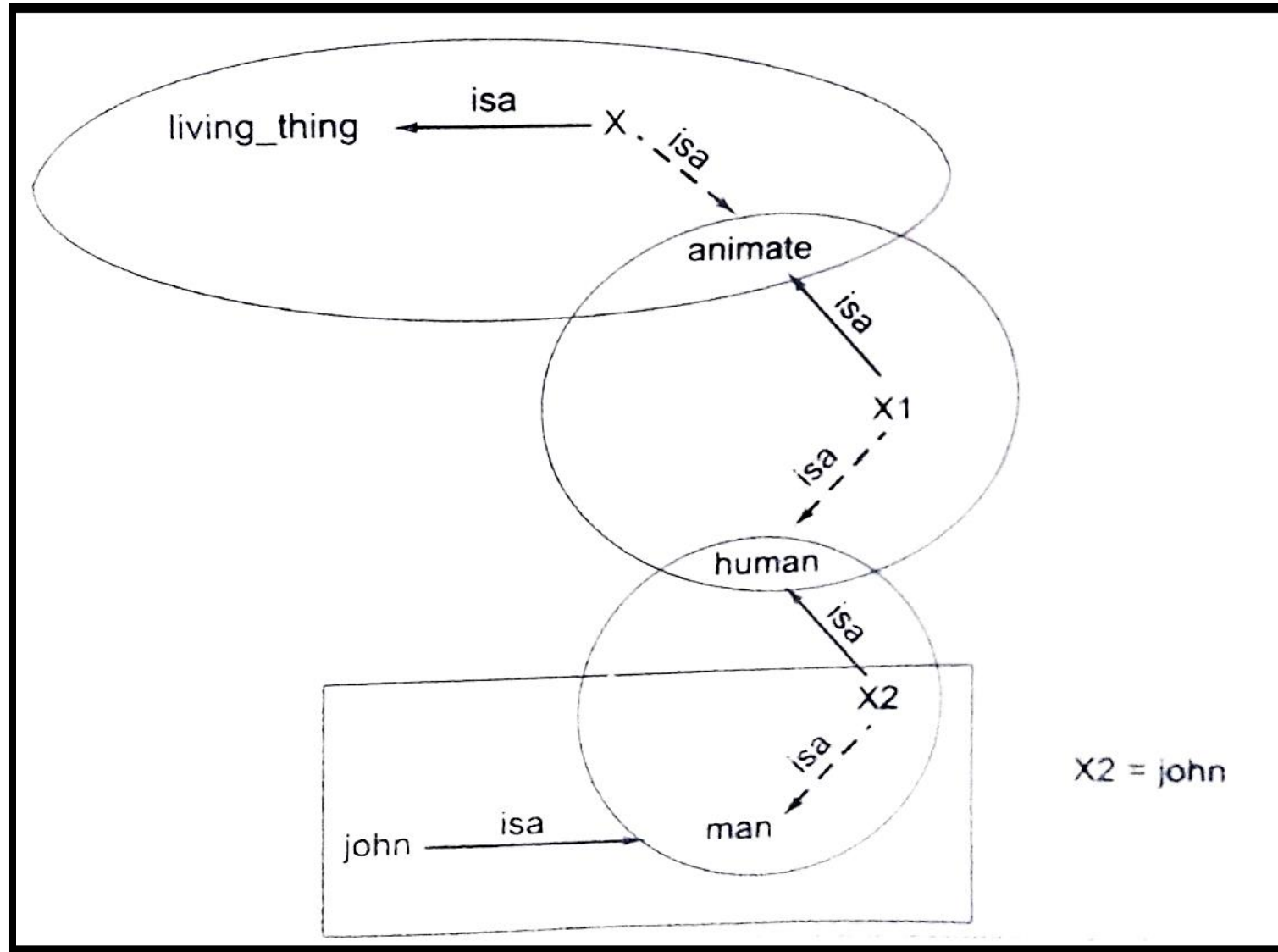
$\text{isa}(X, \text{human}) \leftarrow \text{isa}(X, \text{man})$

$\text{isa}(\text{John}, \text{man})$

ESNet Representation

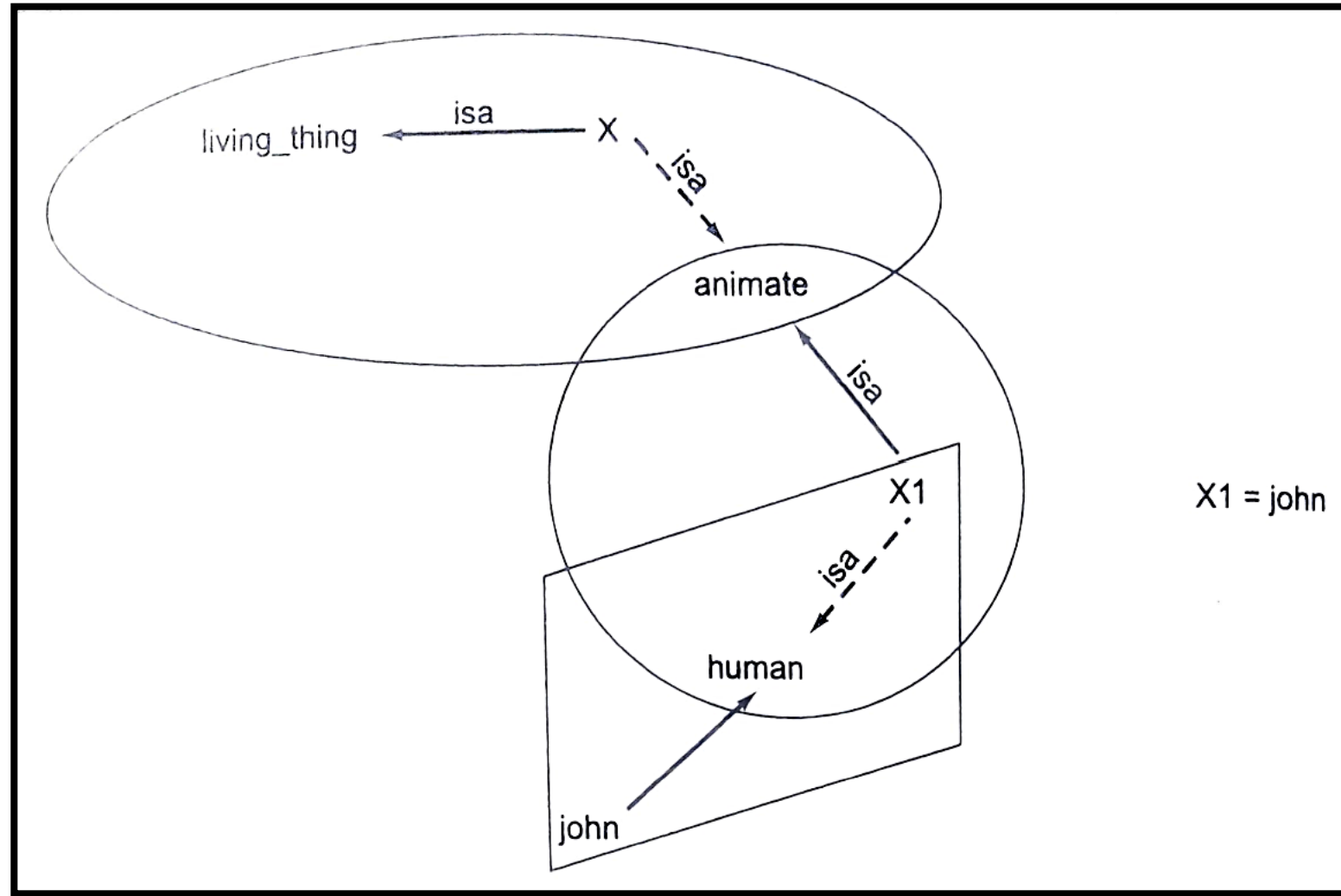


Forward Reasoning Inference



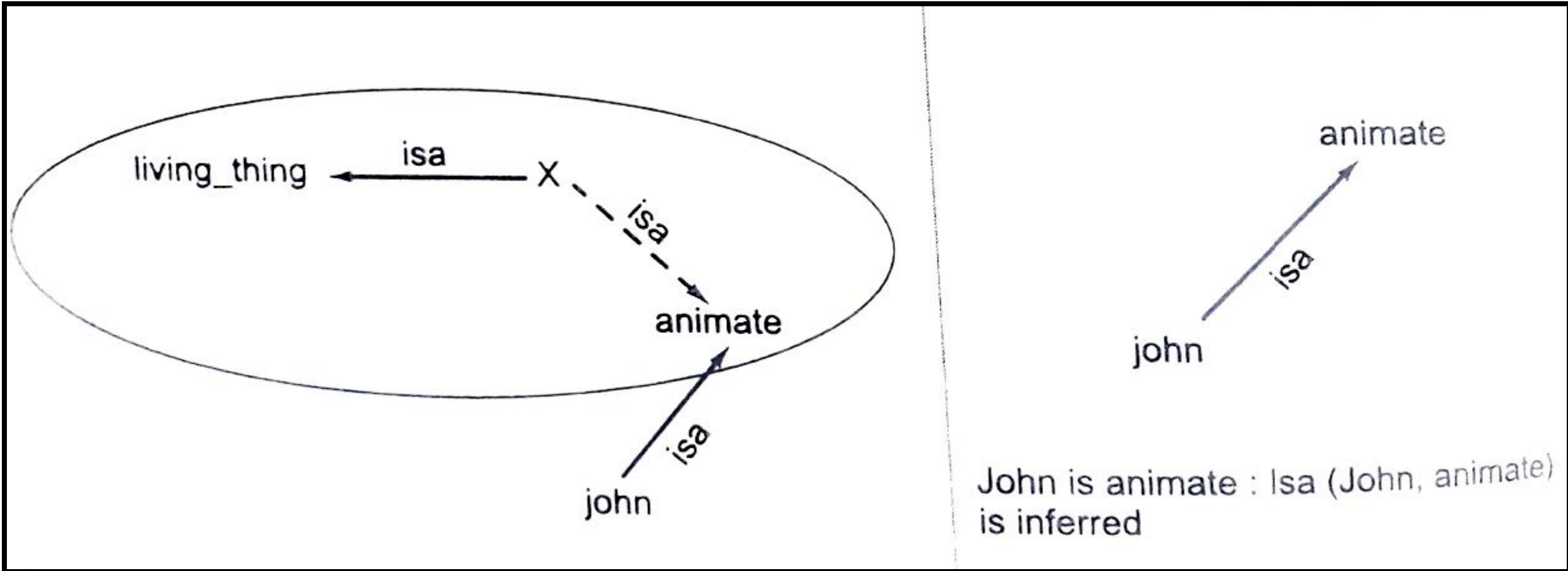
(i)

Forward Reasoning Inference



(ii)

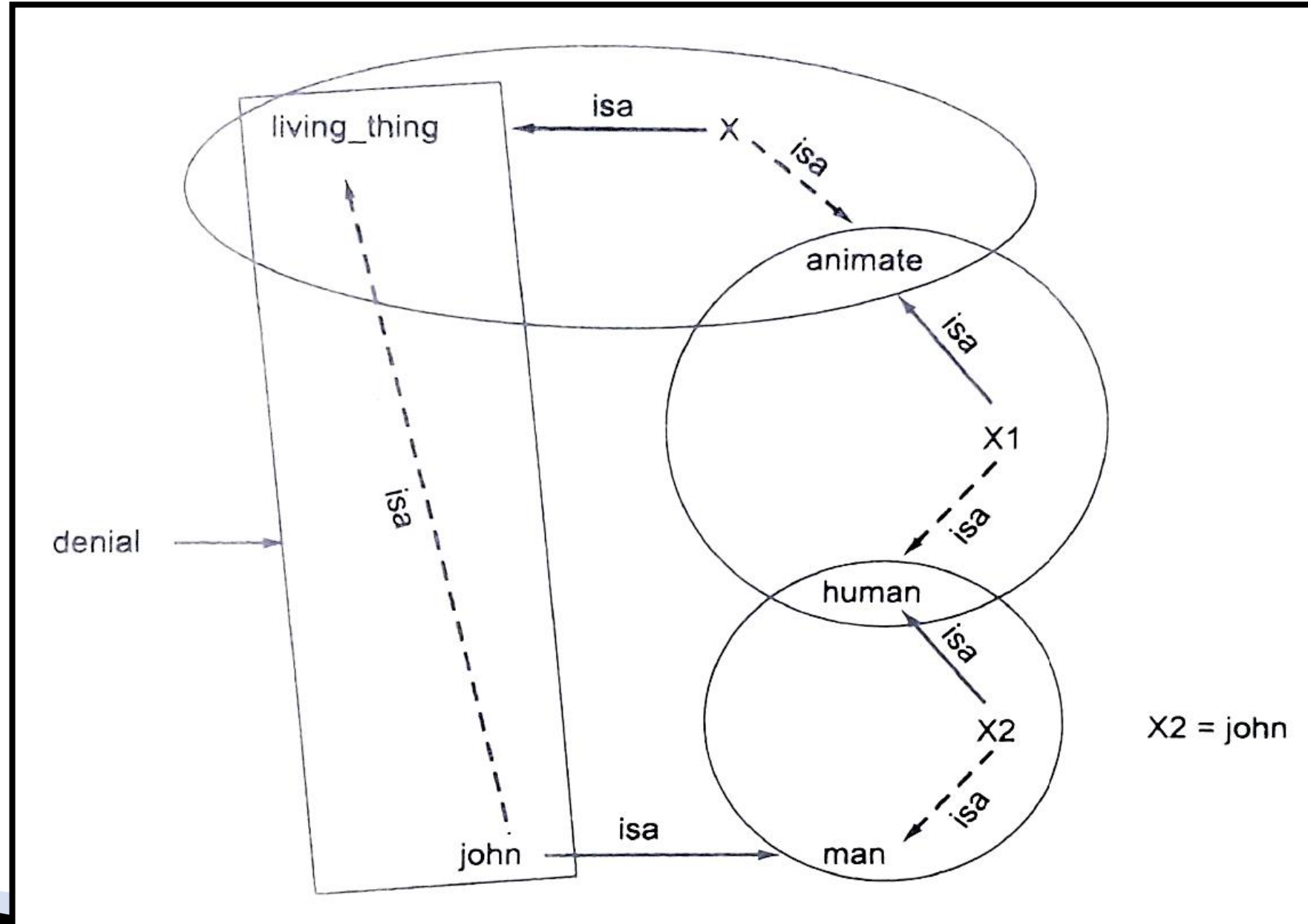
Forward Reasoning Inference



(iii) Derivation of John is animate

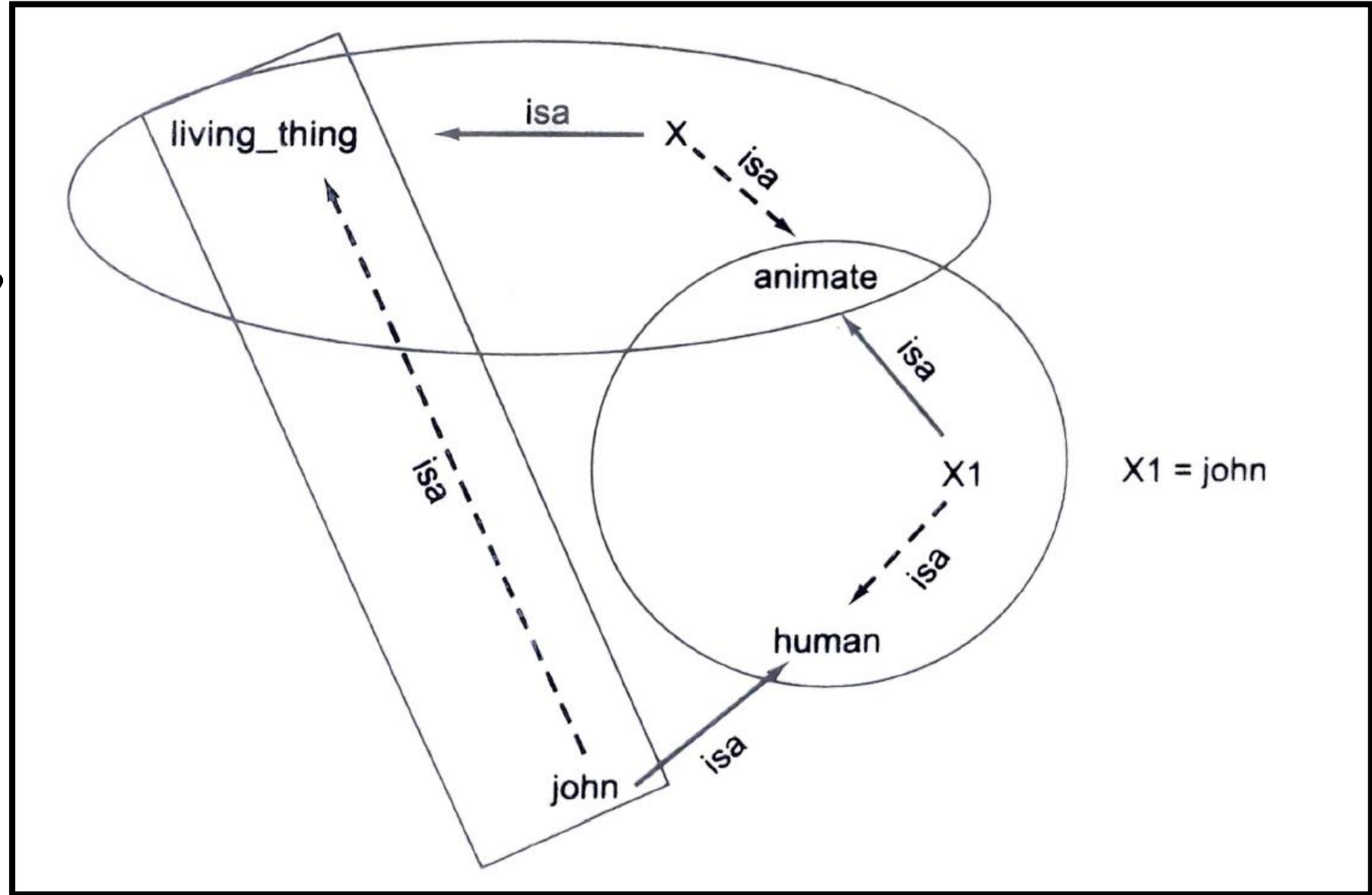
Backward Reasoning Inference

isa(John, living_thing)
i.e. Is John a living thing?



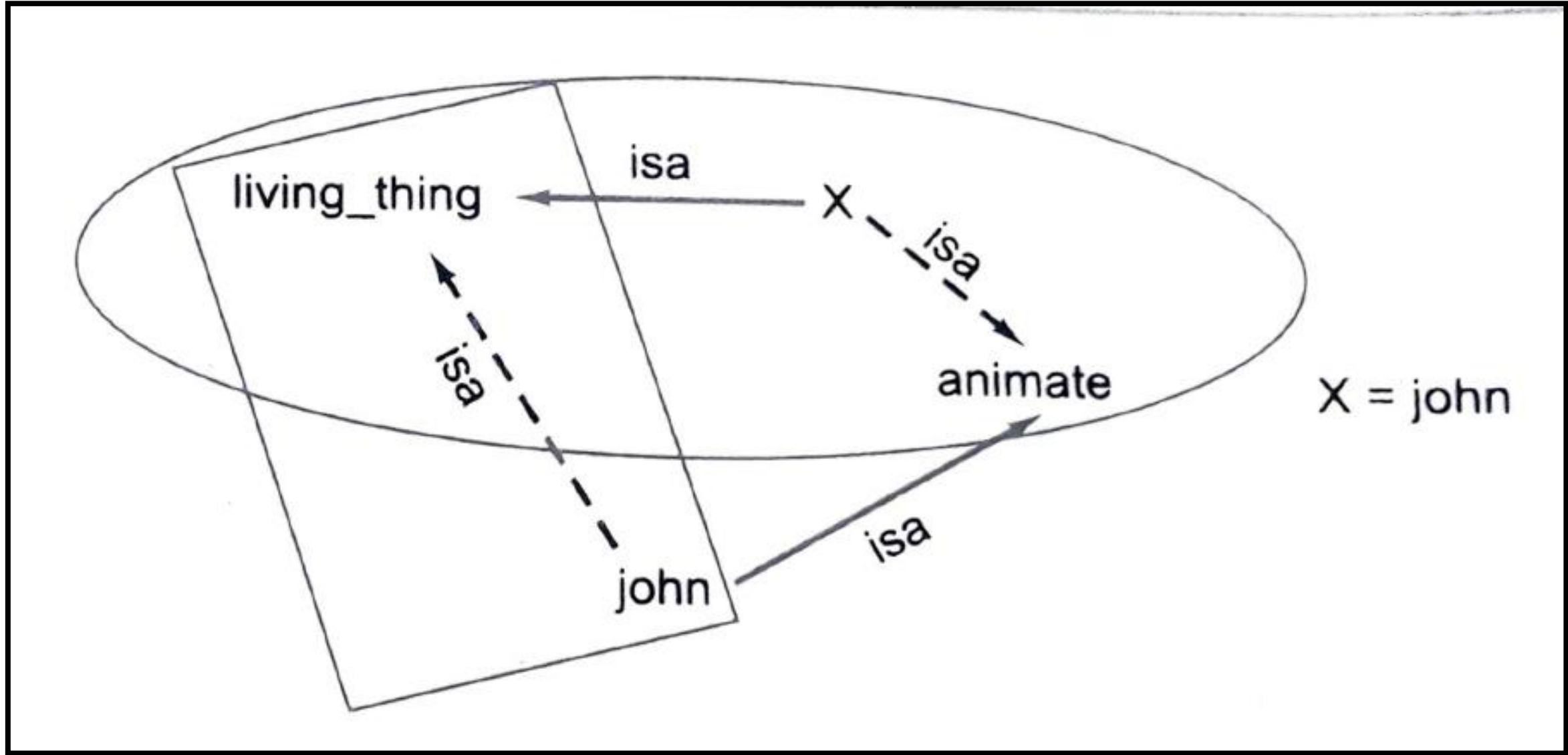
Backward Reasoning Inference

isa(John, living_thing)
i.e. Is John a living thing?

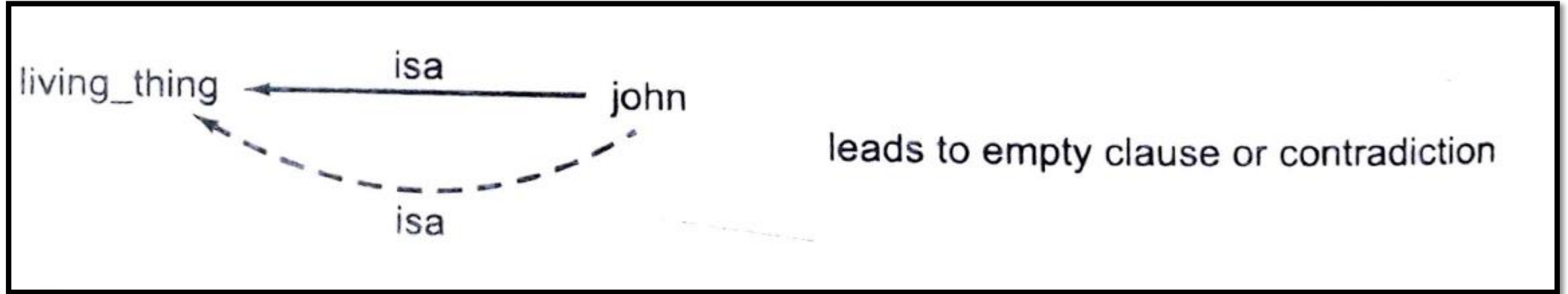


Backward Reasoning Inference

isa(John, living_thing) i.e. Is John a living thing?



Backward Reasoning Inference



Solving query `isa(John, living_thing)`

Example

“Anyone who gives something he likes to a person likes that person also. John gives an apple to Mike. John likes an apple”

The clausal representation:

likes(X, Z) \leftarrow action(E, give), object(E, Y), actor(E, X), recipient(E, Z), likes(X, Y)

action(e, give)

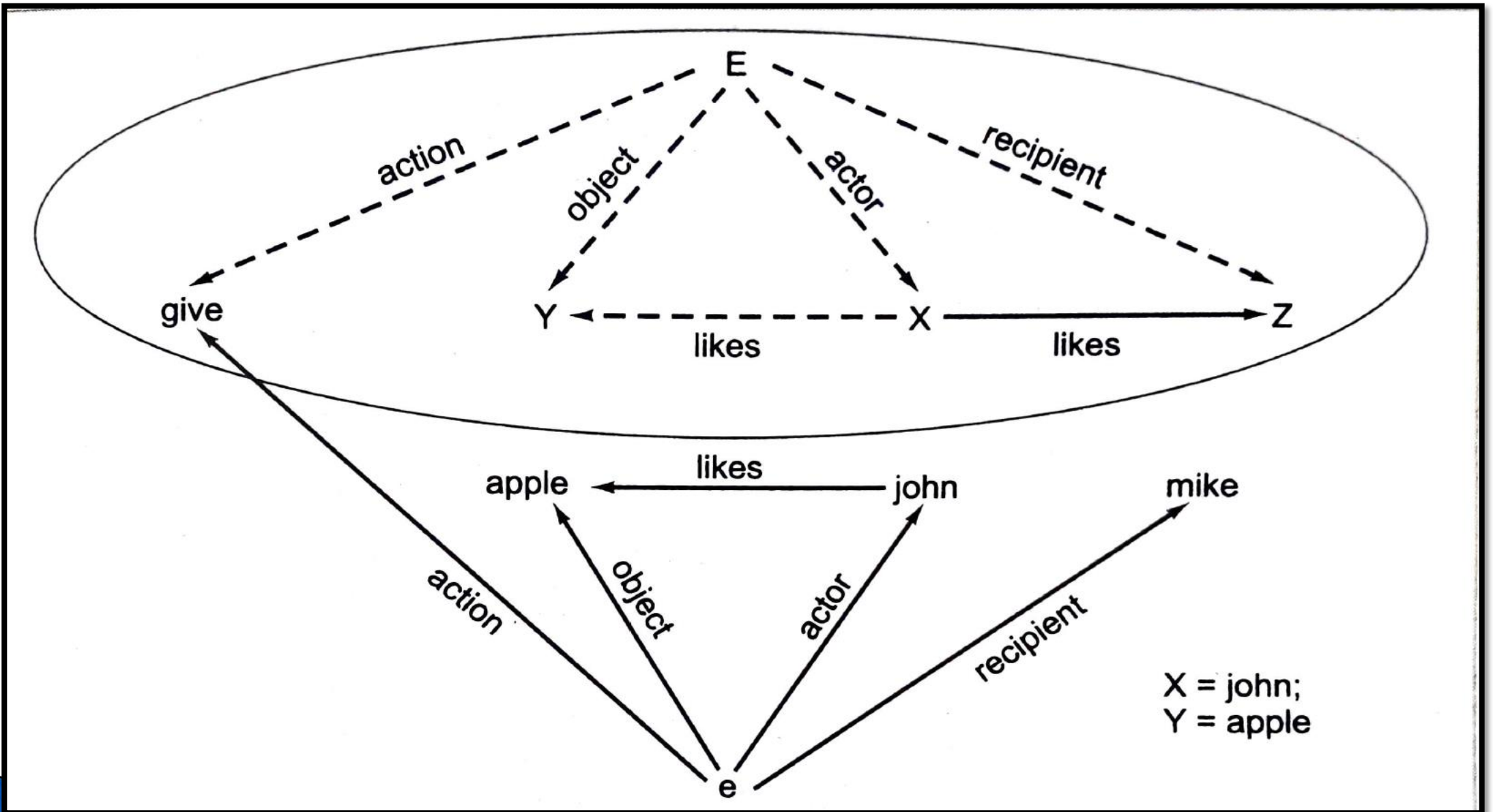
object(e, apple)

actor(e, John)

recipient(e, Mike)

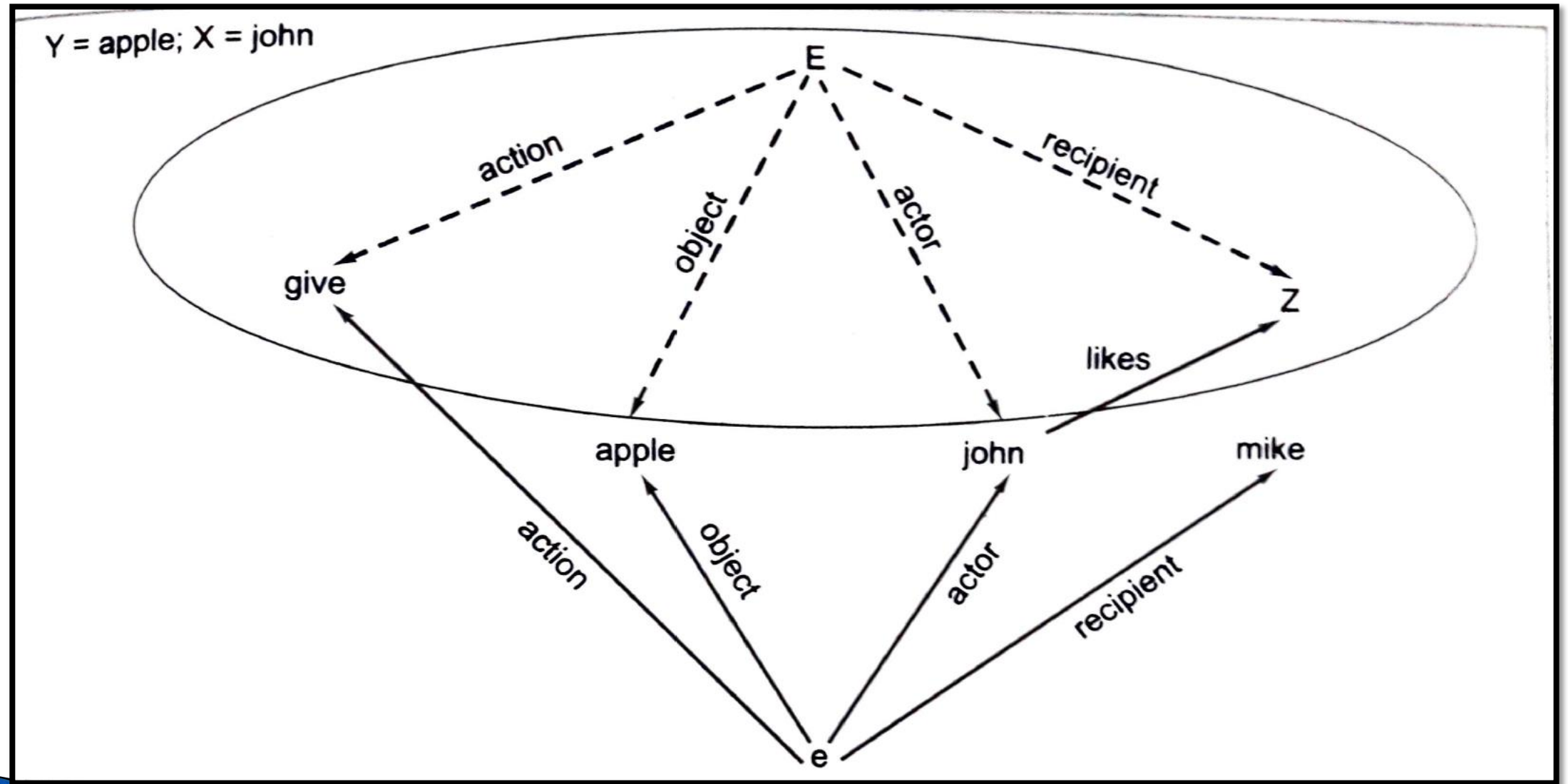
likes(John, apple)

Here E is a variable bound to an actual event ‘e’ in the process of resolution



ESNet Representation

Forward Reasoning Inference



Reduced ESNet

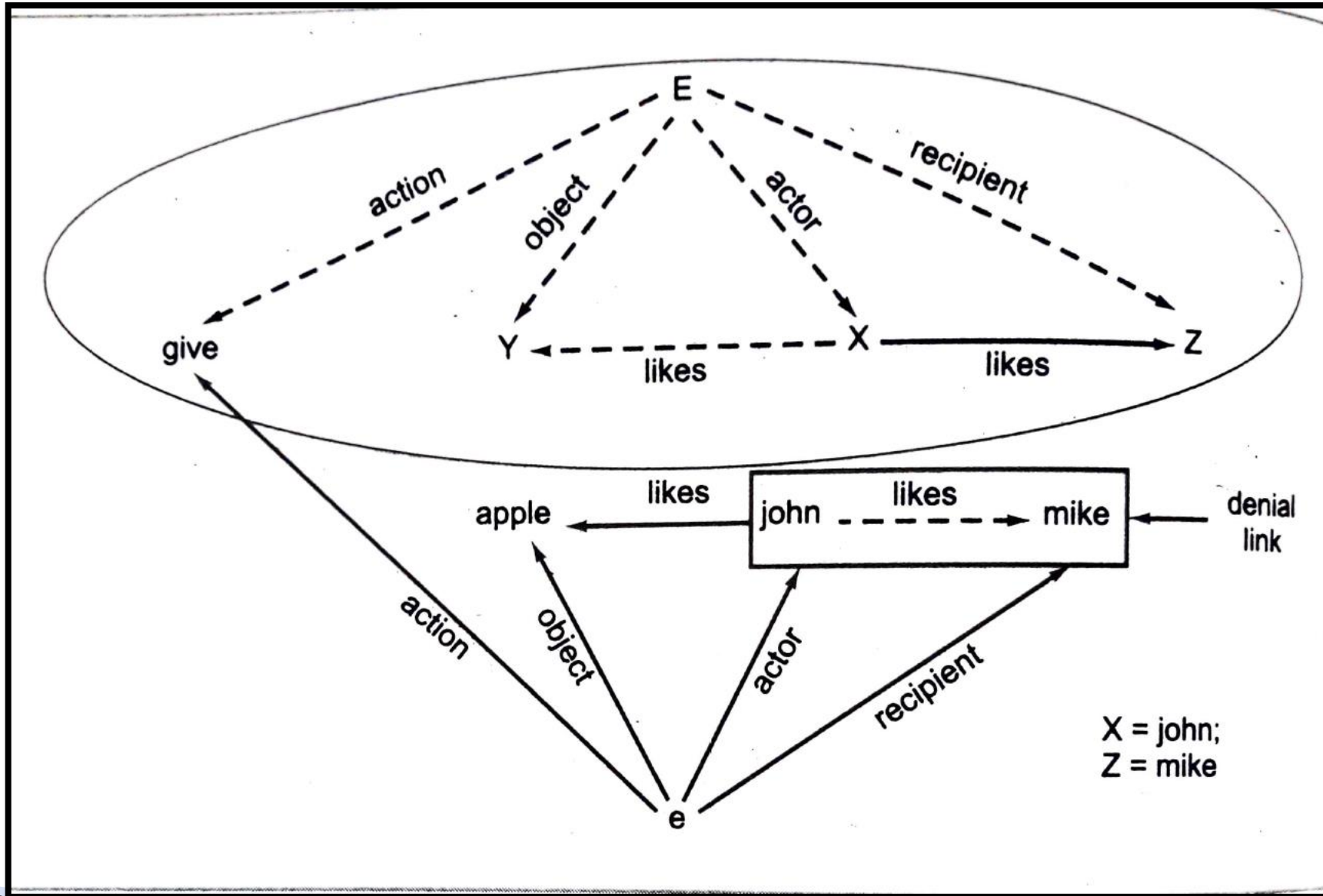
Forward Reasoning Inference

$E=e; Z=Mike$

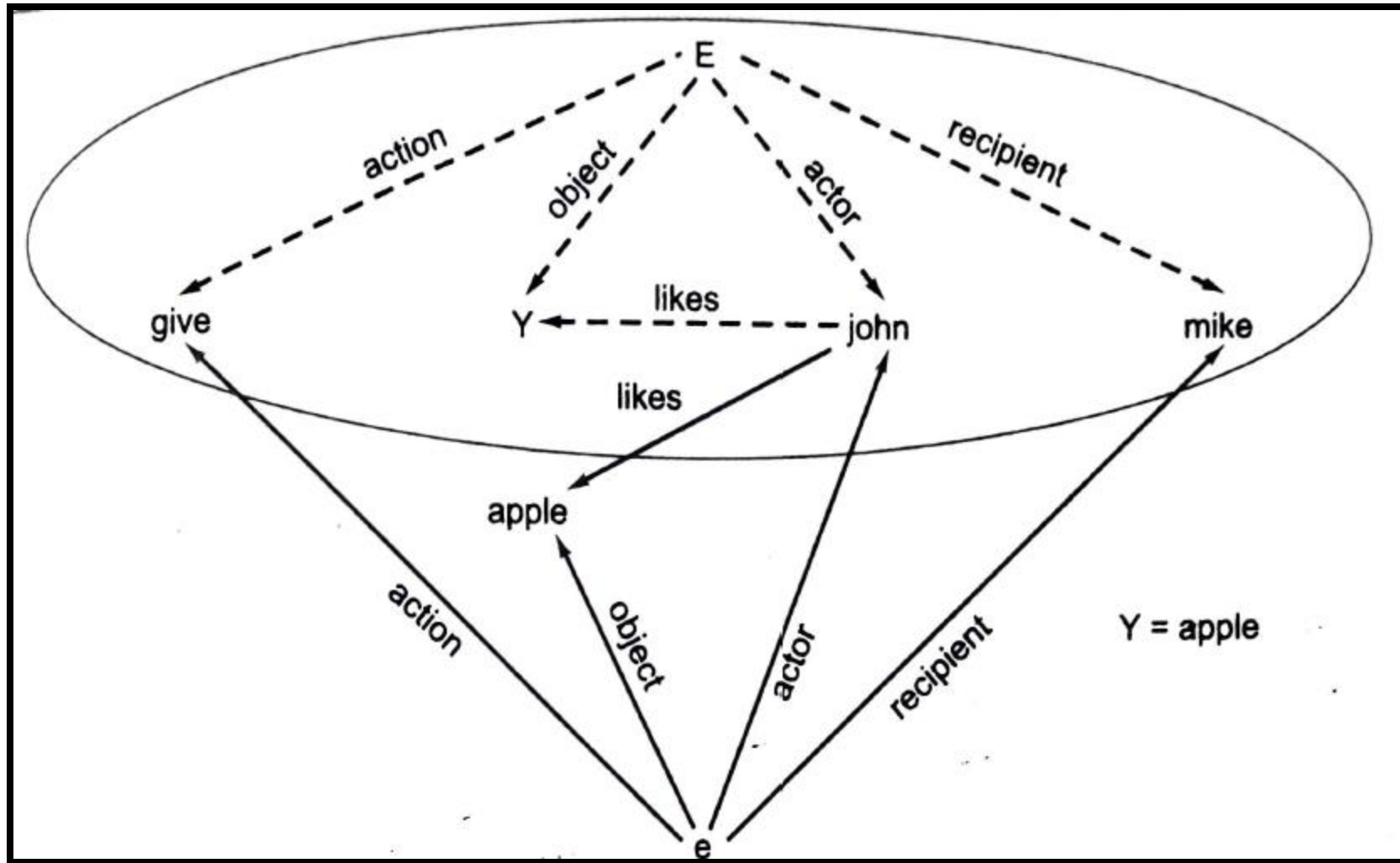
John $\xrightarrow{\text{likes}}$ Mike

Hence proved that : **likes(John, Mike)**

Backward Reasoning Inference

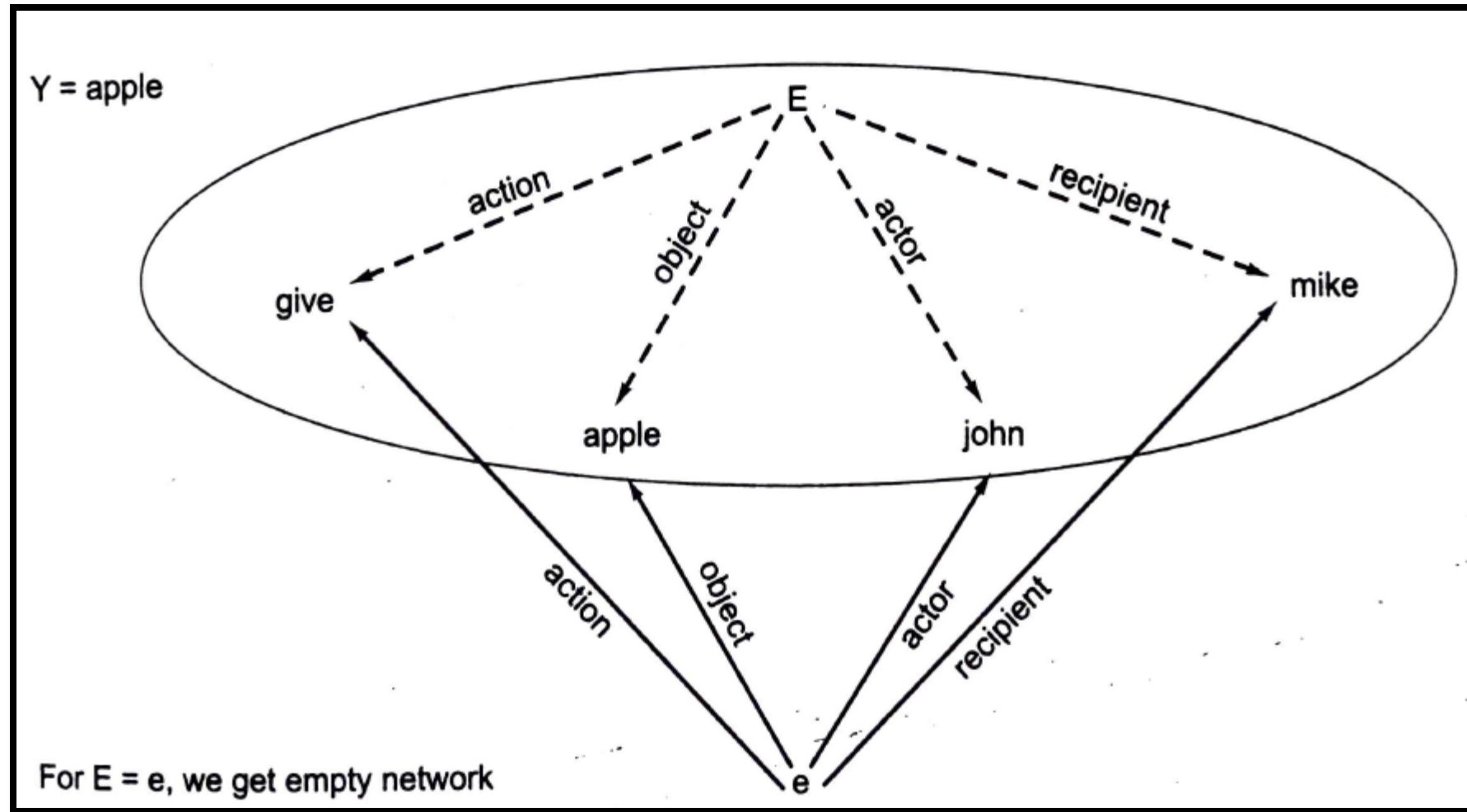


Backward Reasoning Inference



(ii)

Backward Reasoning Inference



(iii) Reduction to Empty ESNet

Inheritance

isa(X,living_thing) \leftarrow isa(X,animate)

isa(X,animate) \leftarrow isa(X,human)

isa(X,human) \leftarrow isa(X,man)

isa(John,man)

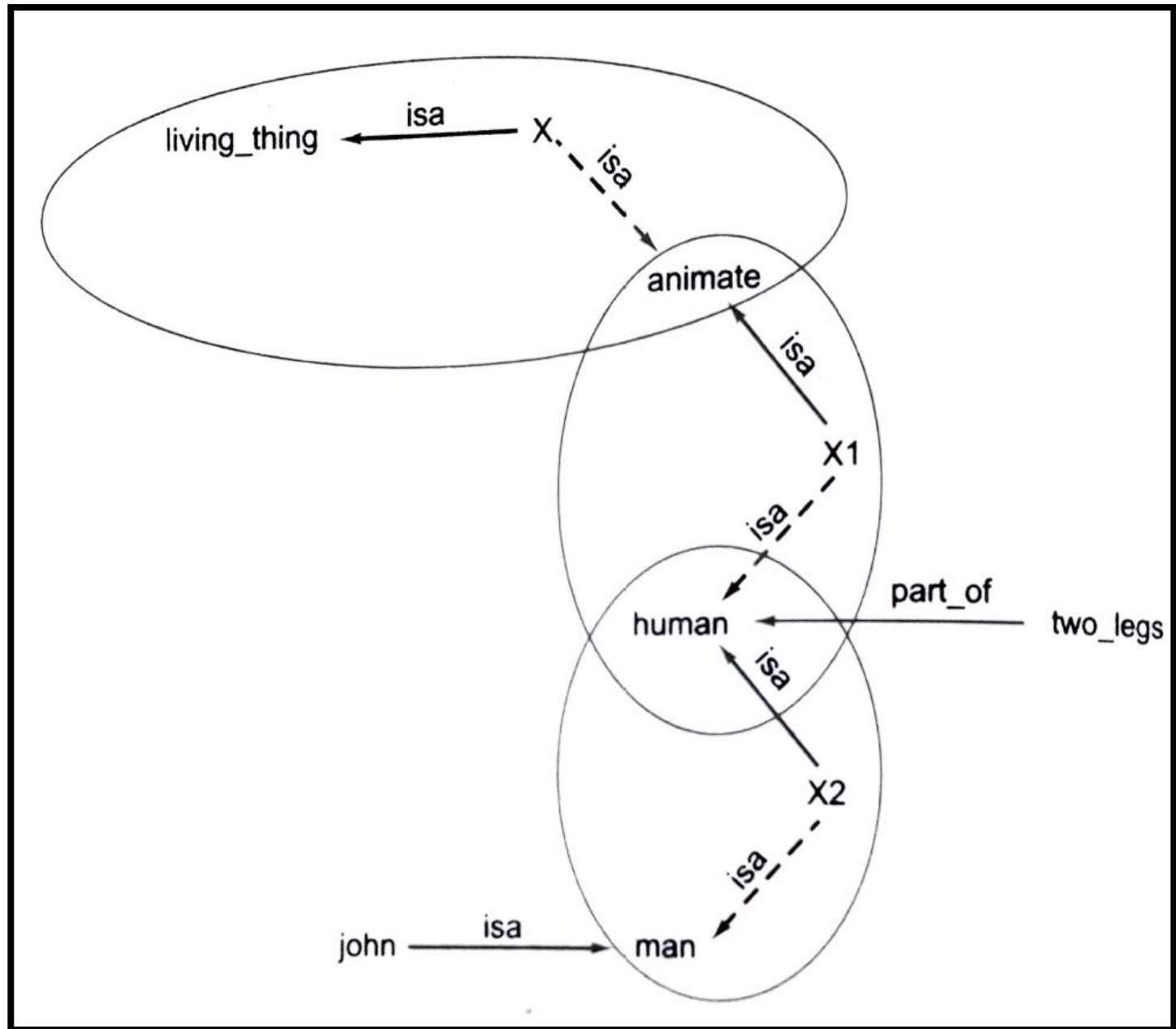
part_of(human, two_legs)

Inheritance

Prove that:

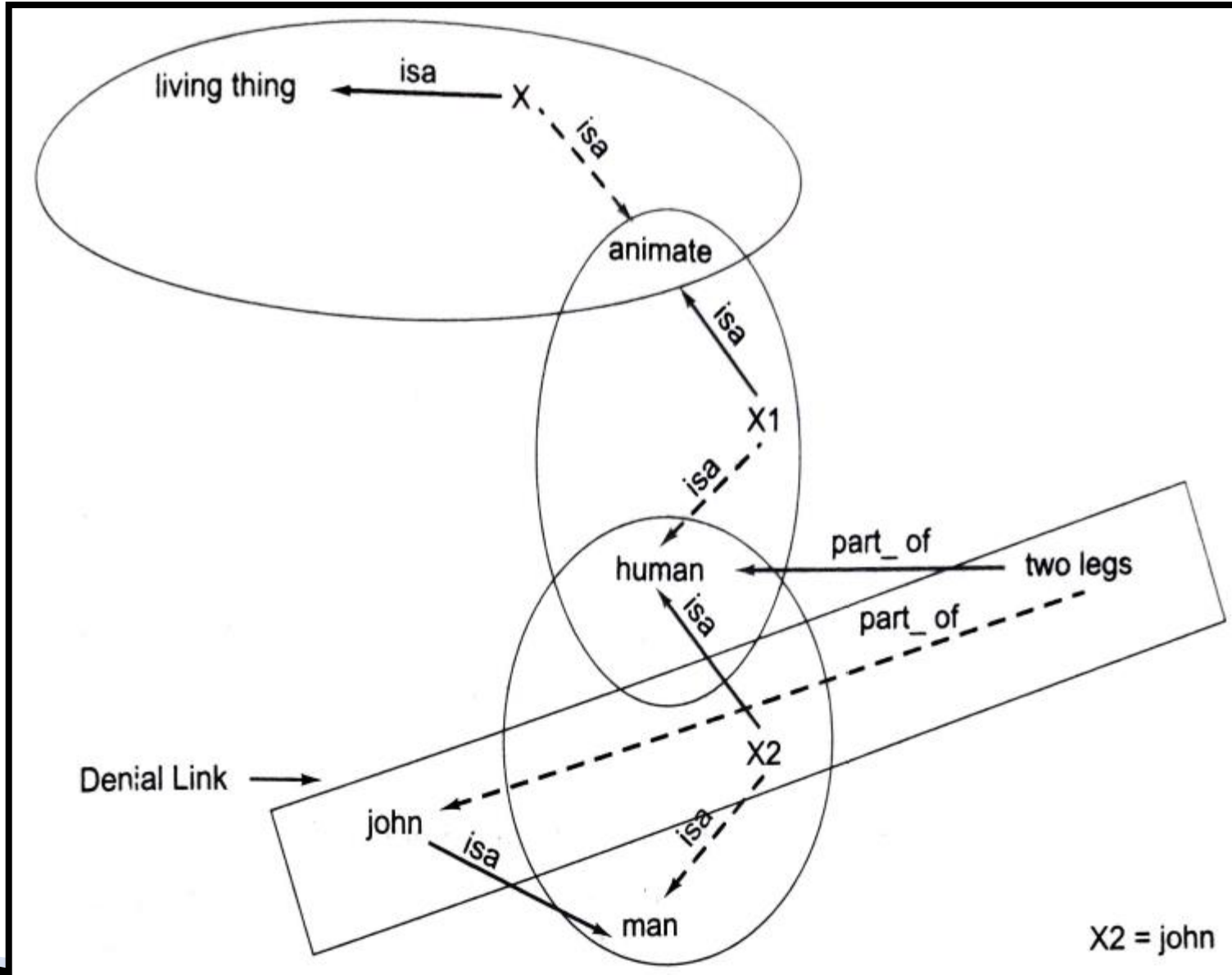
John has two_legs

i.e. `part_of(John, two_legs)`



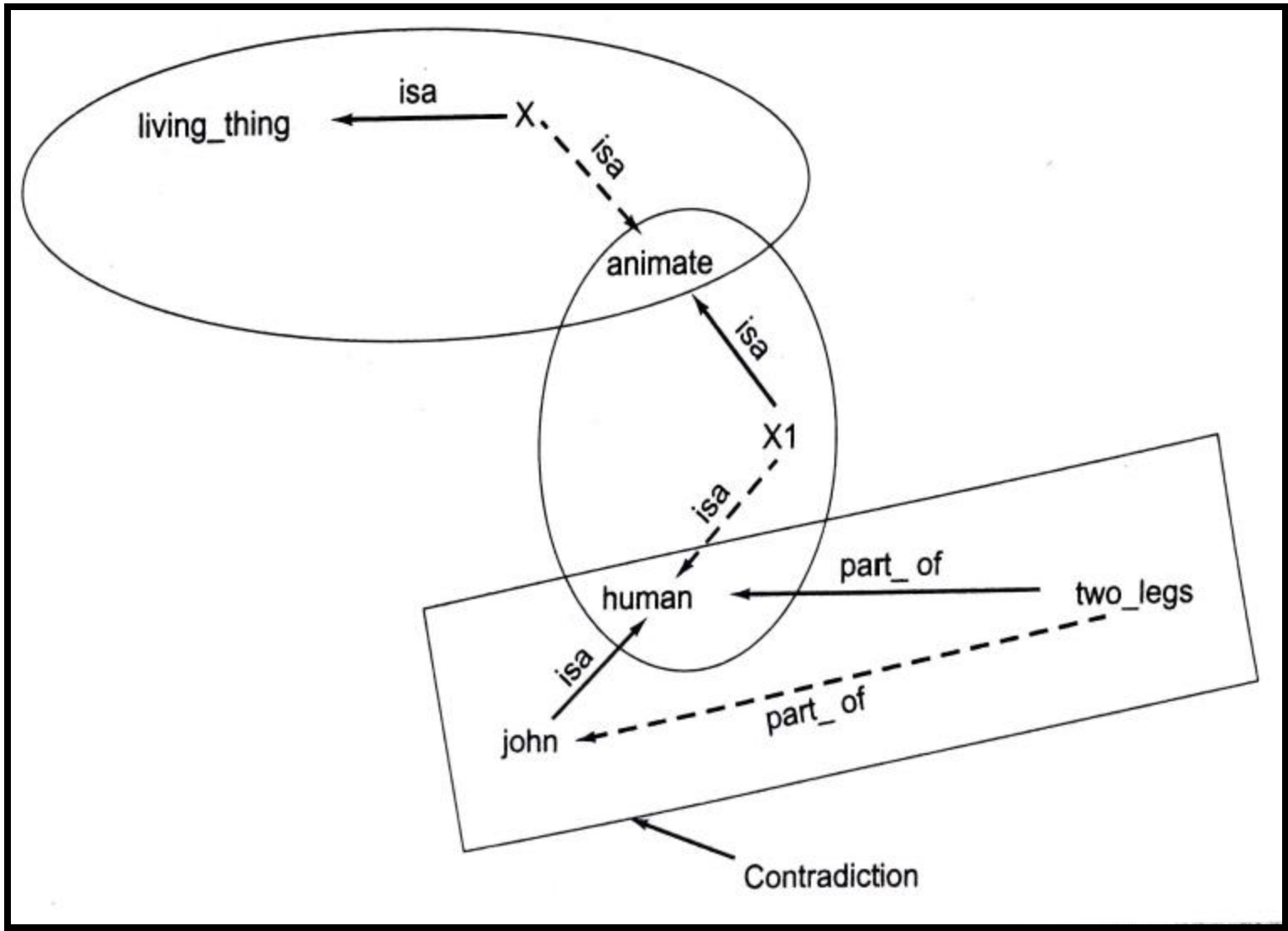
ESNet for Inheritance

Backward Reasoning Inference




Addition of a Denial Link to the network

Backward Reasoning Inference

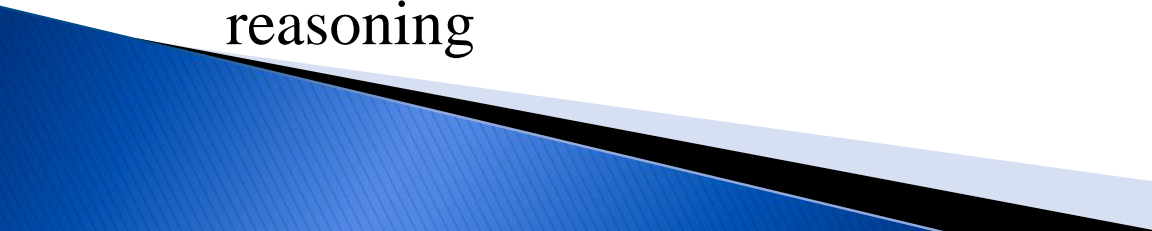


Obtaining a Contradiction

Implementation of ESNet

- Implementation of ESNet can be done in any programming language or using a tool which facilitates implementation of Semantic Networks
 - The clauses can be represented either explicitly by adding them to the network ,or implicitly by using the structure-sharing method
 - In a Conventional Semantic Network, procedures are generally written in the host programming language whereas in ESNet, procedures are integrated with the rest of the database and are executed by the same general purpose mechanism which performs inference in the network
- 

Knowledge Representation using Frames

- Frames are regarded as an extension to semantic nets; each node of a semantic net is represented by a frame
 - A Frame may be defined as a data structure that is used for representing a stereotyped situation
 - It consists of a collection of attributes or slots and associated values that describe some real-world entity
 - There are several types of information attached to each frame
 - Frame systems form a powerful way of encoding information that supports reasoning
- 

Knowledge Representation using Frames

- Frames may be considered to represent the ways of organizing as well as packaging knowledge in a more structured form
- Frames are slightly similar to the concept of class of object-oriented paradigm
- Frames consists of attributes or slots. Slots are described with attribute-value pairs $\langle \text{slot_name}, \text{value} \rangle$
- Slots are generally complex structures that have facets (or fillers) describing their properties. Value of a slot maybe a primitive, such as a text string, constant or an integer, or it may be another frame
- Slots may contain value, refer to other frames(relations) or contain methods
- Frames may contain triggers for checking consistency or obtaining updates of other slots

Knowledge Representation using Frames

frame name
slot-filler
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

Structure of a Frame

Knowledge Representation using Frames

- A frame may contain as many <slots-filler> pairs as required to describe an object
- Fillers are also known as Facets
- Each slot may contain one or more facets or fillers from the list given below:

Facet Name	Description
value	value of the slot
default	default value of the slot and 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
other	may contain rules, other frames, semantic net, or any type of other information

List of Facets in a Frame

Knowledge Representation using Frames

- 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, it remains unchanged for all subclasses and instances of that class
- Each frame in the network is either a class frame or an instance frame

Slot Name	Facet Name	Facet value
F_name	value	Metro hospital
Country	default	India
Phone_No	default	23456778
Address	default	abc

Hospital Frame

Knowledge Representation using Frames

Frames in a network of frames are connected using the links as below:

- **Ako**: This link connects two class frames, one of which is a kind of the other class eg. the class **child_hospital** is a kind of the class **hospital**. A class can define its own slots and also inherits slot-value pairs from its super class. KR becomes more structured and memory efficient
- **Inst** : This link connects a particular instance frame to a class frame eg. **AIIMS** is an instance of the class frame **hospital**. An instance class possesses the same structure as its class frame
- **Part_of**: This link connects two class frames, one of which is contained in the other class eg. **ward** is Part_of the class **hospital**

Network of Frame system for hospitals

Frame Descriptions of some of the frames of the network:

Hospital Frame (Root of the network)

F_name:	hospital
Country:	(value-India)
Phone_No:	(default-23456778)
Address:	(default-New Delhi)
Director:	(default:xyz)
Labs:	lab (Lab Frame)
Wards:	ward (Ward Frame)
Doctors:	doctor (Doctor Frame)

Network of Frame system for hospitals

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)

Lab Frame

F_name: lab
Part_of: hospital (Hospital Frame)
Pathology: pathology (Pathology Frame)
X_Ray: x_ray (X_Ray Frame)

Network of Frame system for hospitals

Ward Frame

F_name: ward

Part_of: hospital (Hospital Frame)

Ortho: orthopaedic (Orthopaedic Ward Frame)

Doctor Frame

F_name: doctor

Part_of: hospital

Qualification:(default-MBBS)

Pathology Frame

F_name: pathology

Part_of: lab (Lab Frame)

Incharge: (default- xyz)

Types of tests:(...)

Network of Frame system for hospitals

X_Ray Frame

F_name: x_ray
Part_of: lab (Lab Frame)
Incharge: (default - pqr)

Orthopaedic Ward Frame

F_name: orthopaedic
Ako: ward (Ward Frame)

Network of Frame system for hospitals

Frame Instance Descriptions

Hospital Frame Instance

F_name:	AIIMS
Inst:	Hospital Frame
Phone_No:	(value - 91 11 6591425)
Address:	(value - Central Delhi)
City:	(value - New Delhi)
Director:	(value – Dr. Smith)
Labs:	(Lab-name)
Wards:	Ward Frame Inst
Doctors:	Doctor Frame Inst

Network of Frame system for hospitals

Frame Instance Descriptions

Lab 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

Pathology Frame Instance

F_name: pathology-lab-name
Inst: pathology (Pathology Frame)
Part_of: lab-names (Lab Frame Inst)
Incharge: (value- Mr John)
Types of tests:(value – [ECG, Blood Test])

Network of Frame system for hospitals

Frame Instance Descriptions

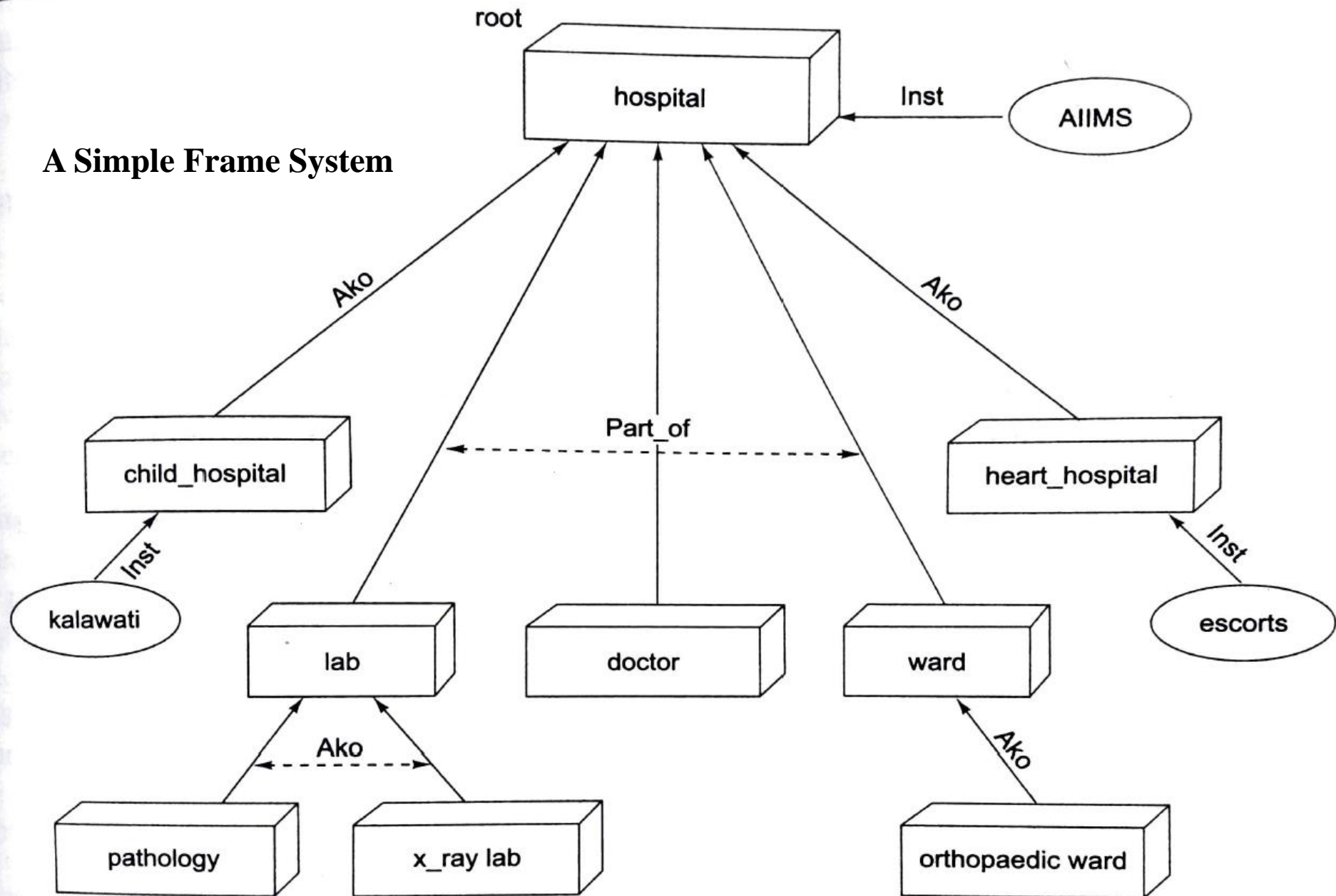
Child Hospital Frame Instance

F_name: Kalawati
Inst: Child_hospital (Child Hospital Frame)
Address: (value – 1234 New Delhi)
Phone_No: (value - 011 6591425)
Director: (value – Dr.Mike)


Heart Hospital Frame Instance

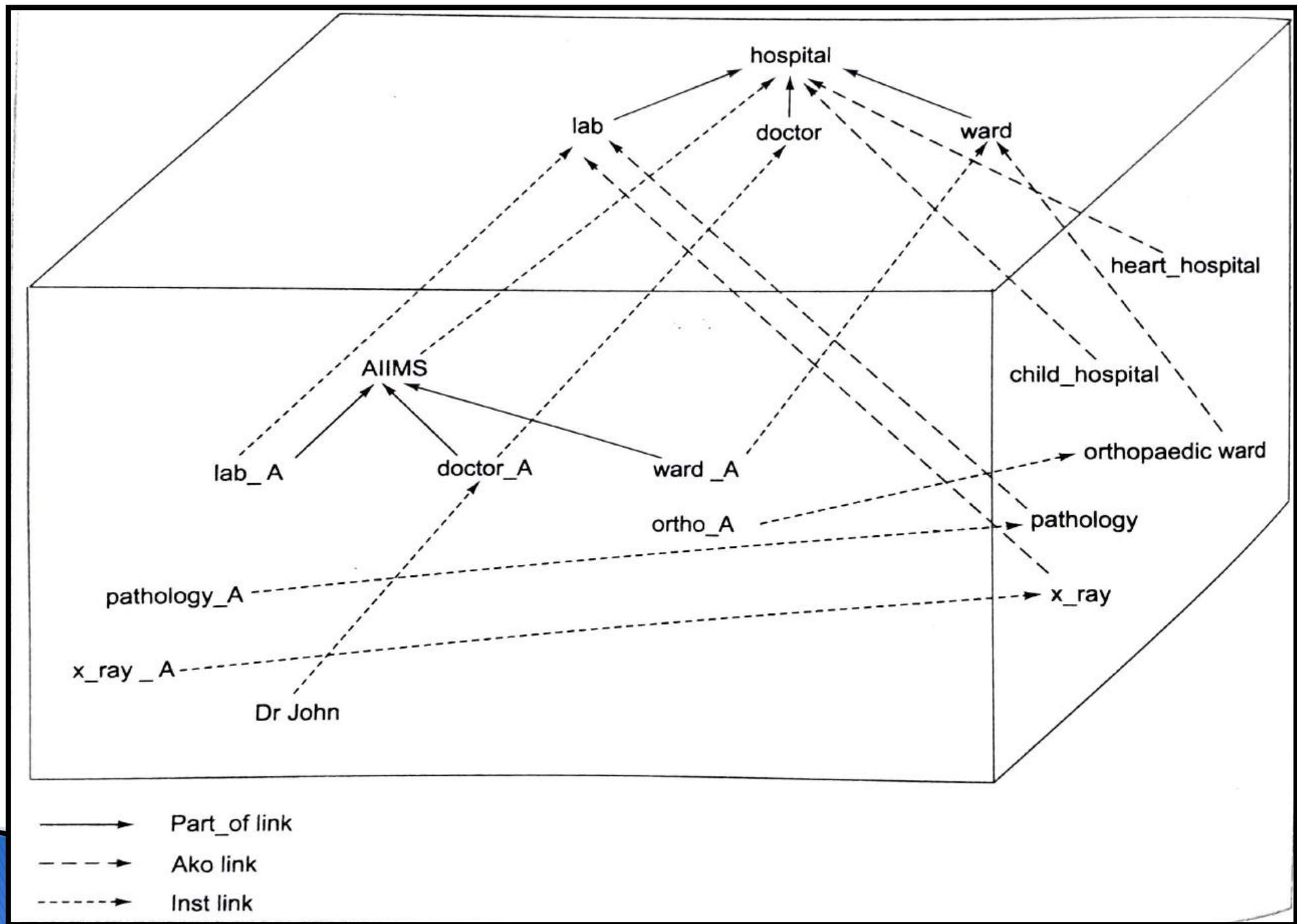
F_name: escorts
Inst: heart_hospital (Heart Hospital Frame)
Phone_No: (value-0114563732)
Address: (value-Faridabad)

A Simple Frame System



Knowledge Representation using Frames

- A network of frames can be viewed as a three-dimensional figure
 - One dimension is concerned with the description of a class along with all the classes contained in it and connected with Part_of link
 - Second dimension in the hierarchy is of Ako links
 - Third dimension contains the instances of all classes connected with Inst links
 - It is important to note that each frame can have atmost two links:
 - Inst and Part_of (Possible)
 - Ako and Part_of (Possible)
 - **Inst and Ako** (Not possible)
- 




Knowledge Representation using Frames

- An instance of a hospital is a specific hospital, such as AIIMS, which will have instances of all the classes in the network of hospital frame, that is, instances of the lab, doctor, and ward
- Representation of the three dimensional structure of a network of frames in first-order predicate logic:

$$\forall X (\text{frame}(X) \equiv \exists Y_1, Y_2, \dots, Y_n ((\text{Ako}(X, Y_1) \vee \text{Inst}(X, Y_1)) \wedge \text{Part_of}(X, Y_2) \wedge \text{slot}_3(X, Y_3) \wedge \text{slot}_4(X, Y_4) \wedge \dots \wedge \text{slot}_n(X, Y_n)))$$

where $\text{frame}(X)$ 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
 - It leads to cognitive economy, where information is only stored in one place
 - **Attachment of Demons(Rules)** : Demons allow 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 eg. An attached demon **if_needed** may be used for monitoring the behaviour of the system
 - **if_added** may be used to validate data when the data is added in the value slot
- 

Inheritance in Frames

- Another use of attaching demons is that they allow dynamic information retrieval and storage
- For eg. A doctor frame can have a slot for tax that contains a demon **if_needed** for calculating tax using salary information available with each doctor instance

Implementation of Frame Knowledge

- Frames can be easily implemented using object-oriented programming (OOP) languages
 - The concept of class in OOP can be directly used to code frames that has the concept of subclass (Ako) and 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 and procedural attachment as methods
 - Inheritance is inbuilt 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(F_name, [slot1(facet(value),...), slot2(facet(value),...),...])`

For eg.

- `frame(hospital, [nil, country(default(India)), phone(default(9111234567)), address(nil), labs(labs) , wards(ward), doctors(doctor)])`
- `frame(child_hospital, [ako(hospital), age(value(0,12), demon(if_needed))])`
- `frame(heart_hospital, [ako(hospital)])`
- `frame(labs, [part_of(hospital), pathology(pathology), x_ray(x_ray)])`

Representation of Frames in Prolog

- `frame(aiims, [inst(hospital), phone(value(91116591425)), address(value(central_delhi)), city(value(delhi)) , labs(labs_inst), wards(ward_inst), doctors([list of doctors])])`
- `frame(labs_inst, [inst(labs), part_of(aiims), pathology(pathology_inst), x_ray(x_ray_inst)])`
- `frame(pathology_inst, [inst(pathology), incharge(value(pqr)), test([ecg, blood_test])])`
- `frame(kalawati, [inst(child_hospital), address(value("1234 New Delhi")), phone(value(0116573891)), director(value(Dr Mike))])`
- `frame(escort, [inst(heart_hospital), phone(value(0114563732)), address(value(faridabad))])`

Inheritance Rules in Prolog

- `find(X,Y) :- frame(X, Z), search(X, Z), !.`
- `find(X,Y) :- frame(X, [inst(Z) | _]) , find(Y, Z), !.`
- `find(X,Y) :- frame(X, [ako(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 pair and will try to match Y for slot
- If a match is found then its facet value is retrieved, otherwise the process is continued till we reach the root frame