Unit-3 (Part-1)

Knowledge Representation

Introduction to Knowledge Representation

•Humans are best at understanding, reasoning, and interpreting knowledge.

•Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world.

•**But how machines do all these things comes under knowledge representation and reasoning**.

https://www.youtube.com/watch?v=V-O-RFSRe-E

Introduction to Knowledge Representation (Cntd..)

Knowledge representation can be described as :

- Knowledge representation and reasoning (KR, KRR) is the part of AI which concerned with AI agents thinking and how thinking contributes to intelligent behaviour of agents.

- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems.

For Example: i. Diagnosing a medical condition

ii. Communicating with humans in natural language

- It is also a way which describes how we can represent knowledge in AI.

Introduction to Knowledge Representation (Cntd..)

- Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

Definition:

- Knowledge-representation is a field of artificial intelligence that focuses on designing computer representations that capture information about the world that can be used for solve complex problems.

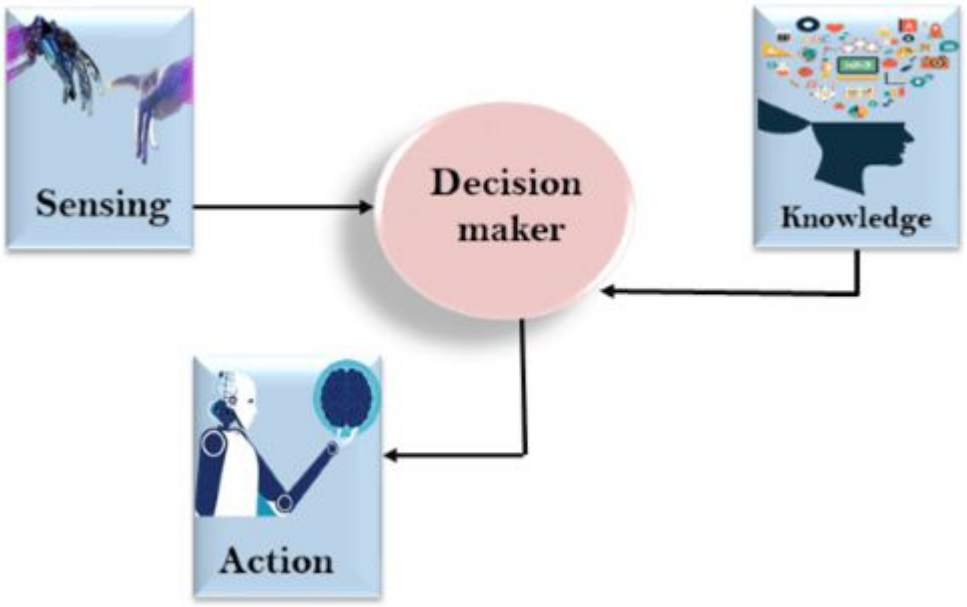
What to Represent:

Following are the kind of knowledge which needs to be represented in AI systems:

- **Object:** All the facts about objects in our world domain.
  E.g., Guitars contains strings, trumpets are brass instruments.
- **Events:** Events are the actions which occur in our world.
- **Performance:** It describe behaviour which involves knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.
- **Facts:** Facts are the truths about the real world and what we represent.

The relation between knowledge and intelligence:

- Knowledge of real-worlds plays a vital role in intelligence and for creating artificial intelligence.
- Knowledge plays an important role in demonstrating intelligent behaviour in AI agents.
- An agent is only able to accurately act on some input when it has some knowledge or experience about that input.



Introduction to Knowledge Representation (Cntd..)

A good knowledge representation system must possess the following properties:

i. **Learning (or) Inferential Efficiency**
ii. **Efficiency in acquisition (or) Acquisitional Efficiency**
iii. **Representational Adequacy**
iv. **Inferential Adequacy**

i. **Learning/Inferential Efficiency –** refers to the capability that acquires new knowledge, behaviours, understanding etc. It does not simple involve adding new facts to a knowledge base but new information may have to be classified to avoid redundancy and replication in the existing knowledge prior to storage to enable easy retrieval.

ii. **Efficiency in acquisition** - the ability to acquire new knowledge using automatic methods wherever possible rather than relying on human intervention.

iii. **Representational Adequacy**-- the ability to represent the required knowledge.

iv. **Inferential Adequacy**- the ability to manipulate the knowledge represented to produce new knowledge from the existing one/original.

Approaches to Knowledge Representation

Basic knowledge representation schemes are:

i. Relational Knowledge
ii. Knowledge Represented as Logic (or) Inferential Knowledge
iii. Procedural Knowledge
iv. Inheritable Knowledge

i. Relational Knowledge:

- Relational knowledge comprises objects consisting of attributes and associated values.
- It is the simplest way of storing facts which uses the relational method.
- Here, all the facts about a set of the object are set out systematically in columns.
- Also, this approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This representation helps in storing the facts, but gives little opportunity for inference.

i. Relational Knowledge:

Example:

A fact: *John is a male whose age is 38 years and who is a graduate with a salary of Rs. 20000.*

Table. Relational Table

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

Queries

- What is the age of John?
- How much does Mary earn?
- What is the qualification of Mike?

Inference: Does a person having a PhD qualification earn more?

ii. Knowledge Represented as Logic

- The inferential knowledge approach represents knowledge in the form of formal logic.
- Thus, it can be used to derive more facts.
- It also guarantees correctness.

Example:

- **Statement 1**: John is a cricketer.
- **Statement 2**: All cricketers are athletes.

It can be represented in predicate logic as:

- **Cricketer(John)**
- **∀x :Cricketer (x) —> Athelete (x)s**

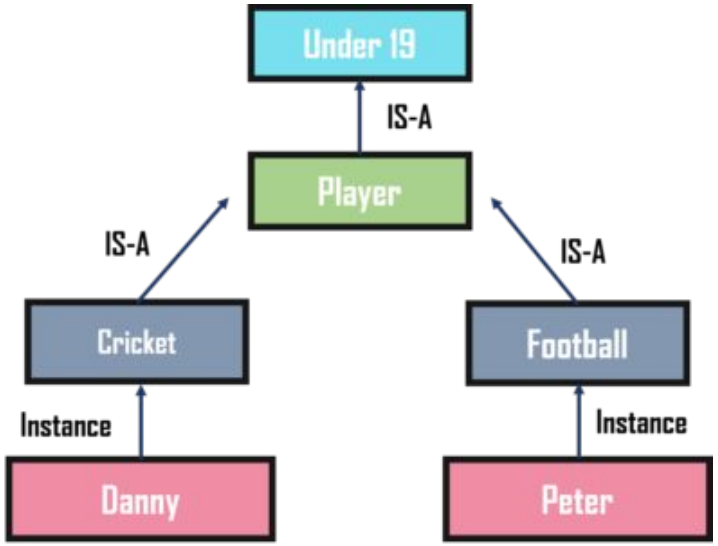
iii. Procedural Knowledge

- Procedural Knowledge also known as Interpretive knowledge, is the type of knowledge in which it clarifies how a particular thing can be accomplished.
- It is not so popular because it is generally not used.
- It emphasize how to do something to solve a given problem.
- Procedural Knowledge is generally process oriented in nature.
- In Procedural Knowledge debugging and validation is not easy.
- Procedural Knowledge is less effective in competitive programming.

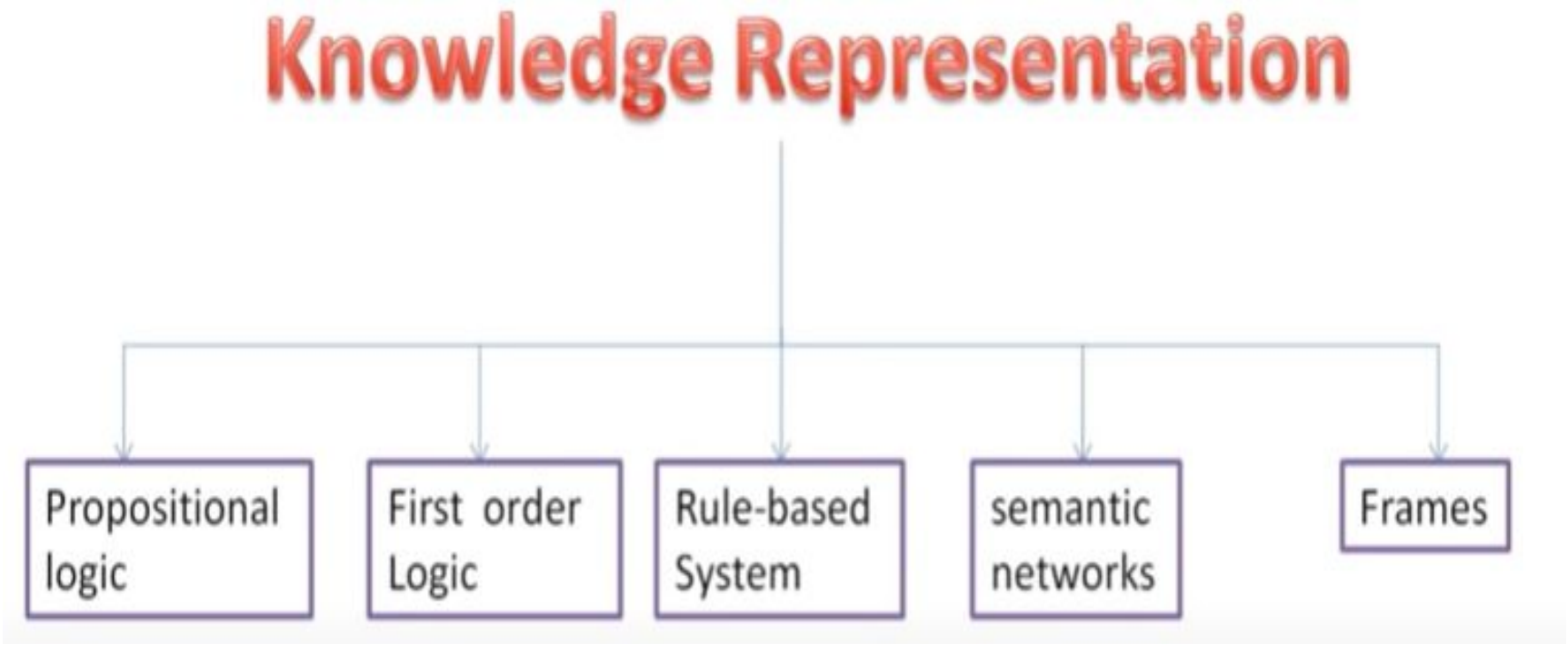
iv. Inheritable Knowledge

- In the inheritable knowledge approach, all data must be stored into a **hierarchy of classes** and should be arranged in a generalized form or a hierarchal manner.
- Also, this approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.

Example:



•Knowledge organized in the form of semantic network and frames is referred as *inheritable knowledge.*



https://www.youtube.com/watch?v=nRMLTHDAVqc

Knowledge Representation using Semantic Network

- In Semantic networks, knowledge can be represented in the form of graphical networks.
- This network consists of nodes representing objects and arcs which describe the relationship between those objects.
- Semantic networks became popular in artificial intelligence and natural language processing only because it represents knowledge or supports reasoning.
- It is an alternative of predicate logic for knowledge representation.
- Semantic networks are easy to understand and can be easily extended.

https://www.youtube.com/watch?v=1jyLrS6U3hA

Knowledge Representation using Semantic Network

- Relationships provide the basic needed structure for organizing the knowledge, so therefore objects and relations involved are also not needed to be concrete.

- Semantic nets are also referred to as associative nets as the nodes are associated with other nodes

- This representation consist of mainly two types of relations:
  - isa relation (Inheritance)
  - inst (Kind-of-relation)

Knowledge Representation using Semantic Network

- **isa**- This relation connects two classes (generic concepts), where one class is a kind or subclass of the other concept.

  Example: Man is a human -> Man is a subclass of the human class

- **Inst –** This relation relates specific members of a class.

  Example: John is an instance of Man

Knowledge Representation using Semantic Network

Example: Following are some statements which need to be represented in the form of nodes and arcs.

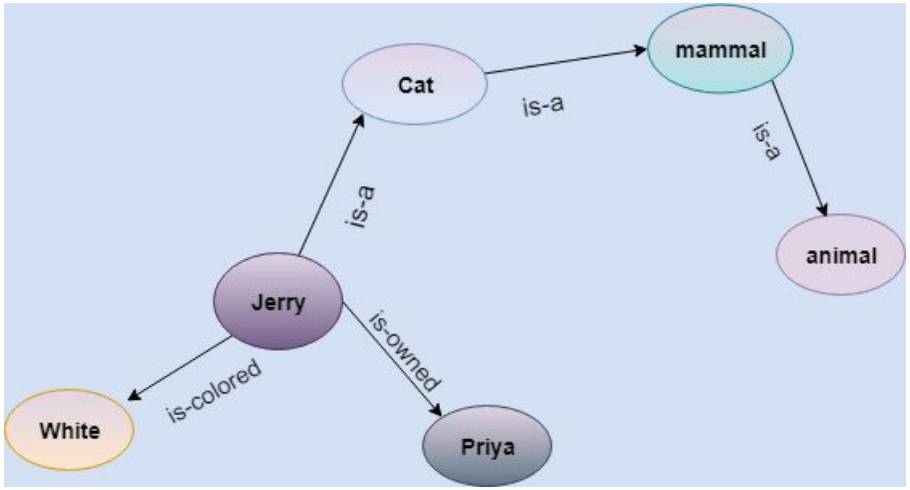
Statements:

i. Jerry is a cat.
ii. Jerry is a mammal
iii. Jerry is owned by Priya.
iv. Jerry is brown colored.
v. All Mammals are animals.
vi. John has an umbrella.

Knowledge Representation using Semantic Network

Statements:

i. Jerry is a cat.
ii. Jerry is a mammal
iii. Jerry is owned by Priya.
iv. Jerry is white colored.
v. All Mammals are animal.



•Represented different type of knowledge in the form of nodes and arcs.
•Each object is connected with another object by some relation.

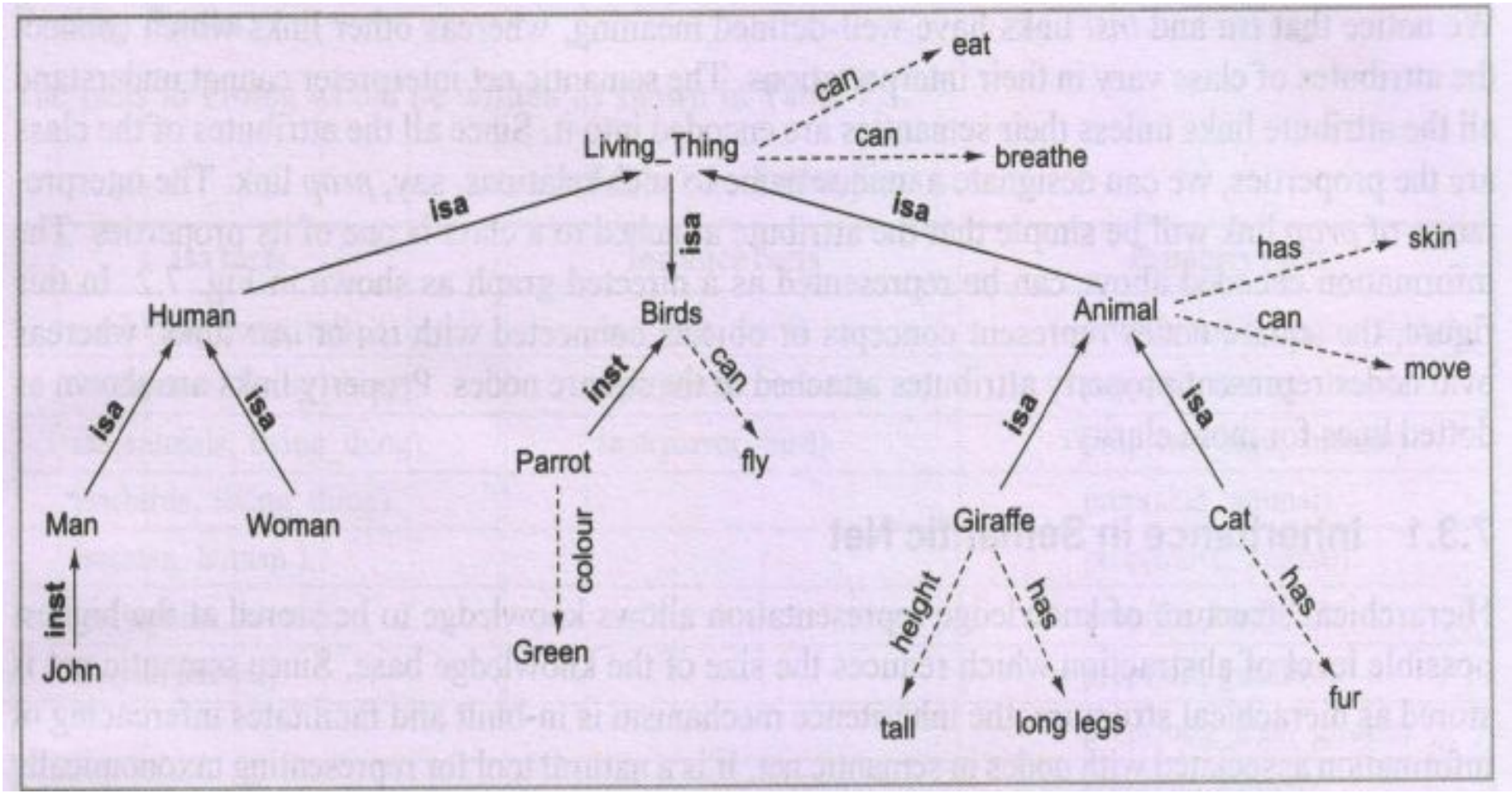


Figure. Knowledge Representation using Semantic Network

- Concepts are related to each other by certain relations.
- These relations are represented in the Figure by bold directed links.
- Other relations such as {can, has, color, height} are known as *property relations*, which are represented by dotted lines pointing from the concept to its property.

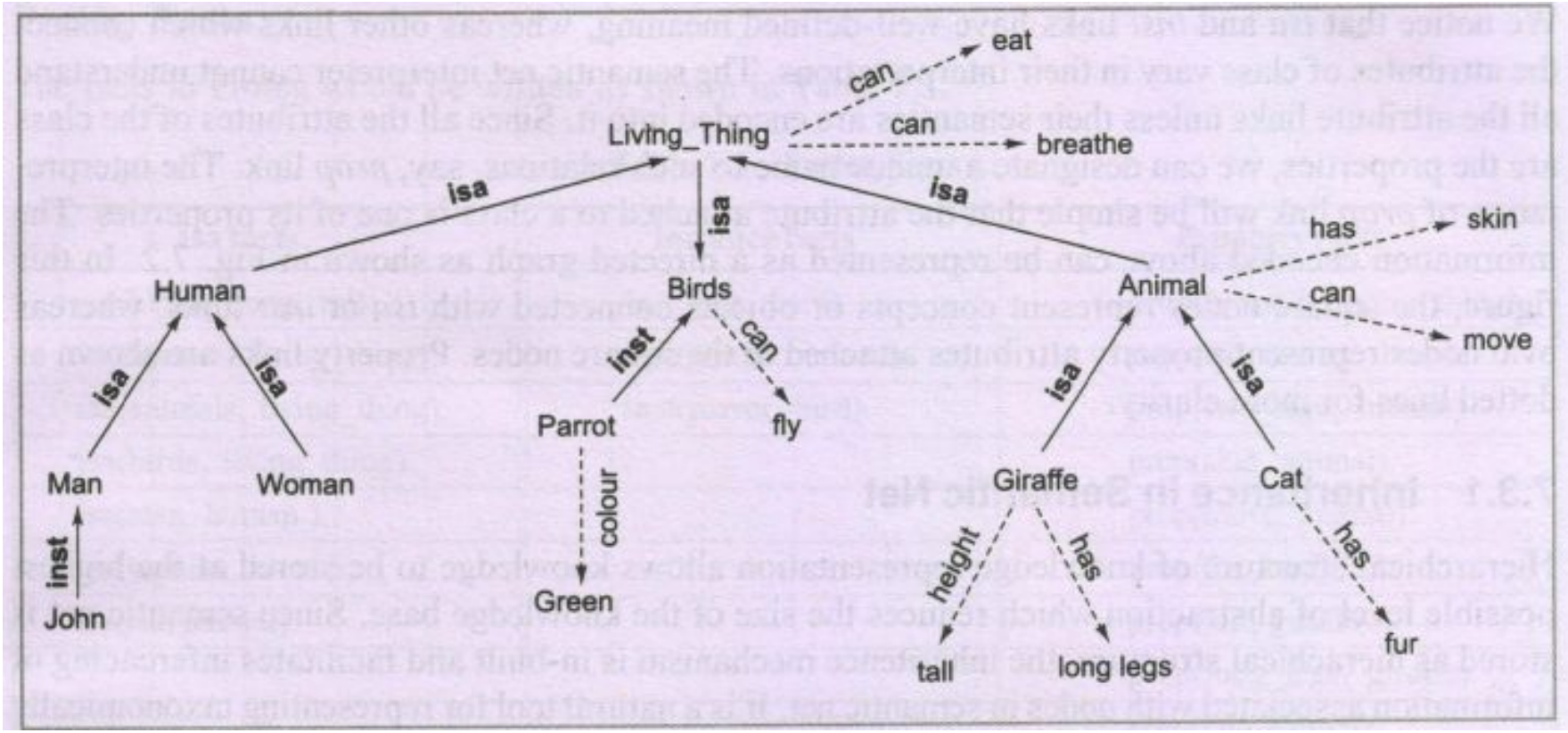


Figure. Knowledge Representation using Semantic Network

- **Query**- *Does a parrot breathe?*

- **Can easily be answered** as '*Yes*' even though this property is not directly associated with a parrot

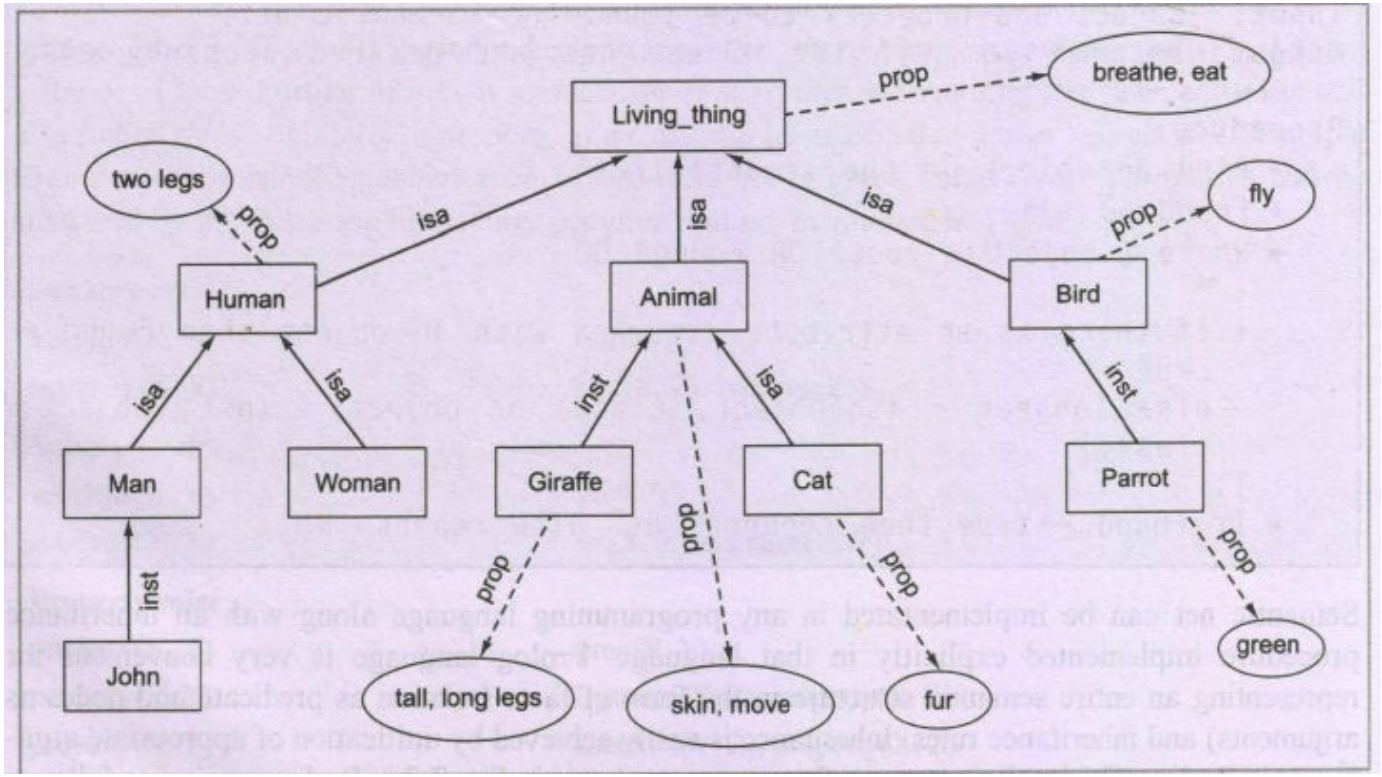


Figure. Concepts connected with *prop* links

•Nodes appear in form of circles or ellipses or even rectangles/squares which represents objects such as physical objects, concepts or situations.

•**Rectangles/square nodes** represents objects connected with i*sa* or *inst* links.

•**Oval/circle nodes** represents property attributes attached to the Rectangles/square nodes .

•***Property links*** are shown as dotted lines for more clarity.

Inheritance in Semantic Net

- Hierarchical representation of knowledge allows knowledge to be stored at the highest possible level of abstraction that reduces the size of the knowledge base.

- 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 helps to maintain consistency of the knowledge base by adding new concepts and members to existing ones.

- Properties attached to a particular concept can be easily inherited by all sub-concepts and their members.

Inheritance in Semantic Net

Algorithm 7.1 *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 ≠ 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

•Semantic net can be implemented in any programming language along with 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 relations.

•Inheritance is easily achieved by unification of appropriate arguments in prolog.

Prolog Facts

Facts are properties of objects, or relationships between objects;

Example:

"Agnibha has phone number 1122334455", is written in Prolog as:

```
phoneno(agnibha, 1122334455).
```

Prolog Facts

- ✓ Names of properties/relationships begin with lower case letters.
- ✓ The relationship name appears as the first term
- ✓ Objects appear as comma-separated arguments within parentheses.
- ✓ A period "." must end a fact.
- ✓ Objects also begin with lower case letters. They also can begin with digits (like 1234), and can be strings of characters enclosed in quotes e.g. color(penink, 'red').
- ✓ phoneno(agnibha, 1122334455). is also called a predicate or clause.

Prolog Facts

Table. Prolog Facts

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).
```

Inheritance Rules in Prolog

Table. 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

• Simple semantic network can only express collections of variable-free assertions.

Example:

John gives an apple to mike and *john and mike are human*.

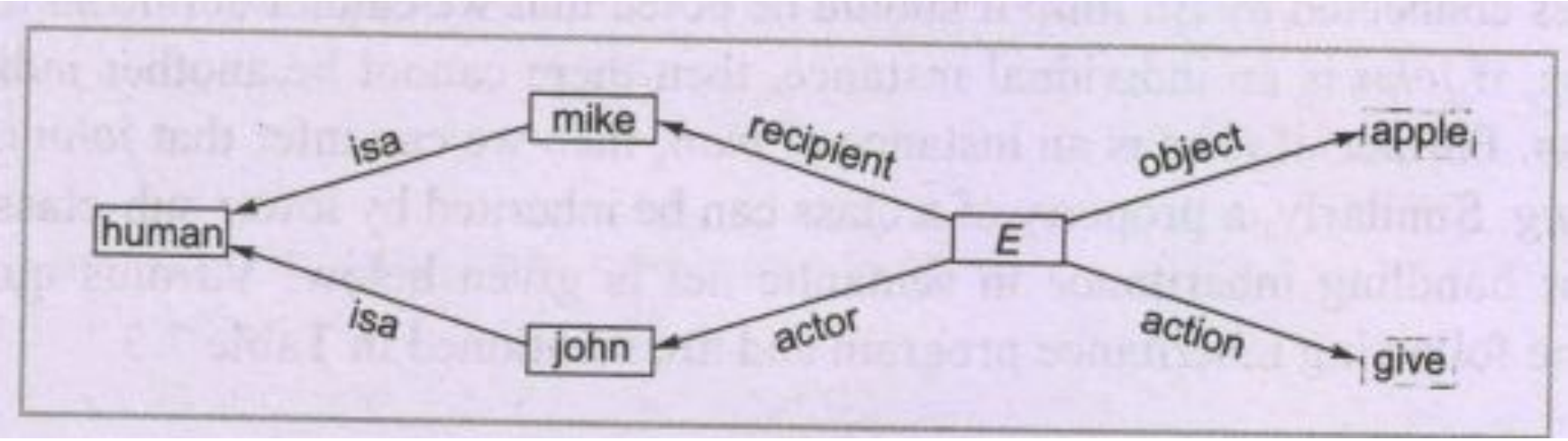


Figure. Semantic Net

• 'E' represents an event which is an act of *giving.*

Extended Semantic Networks for KR

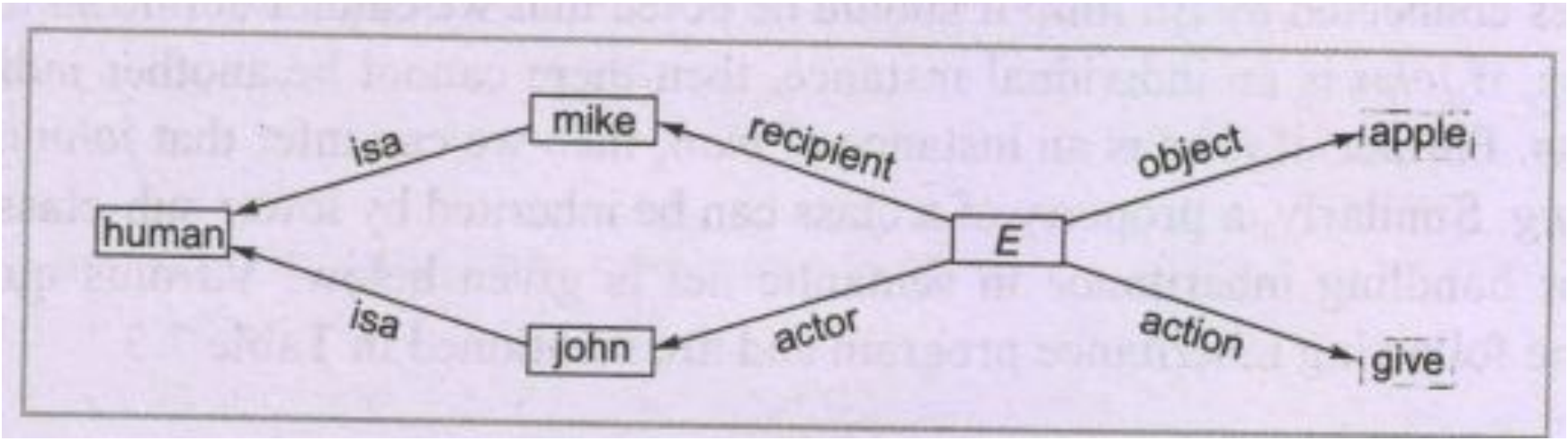


Figure. Semantic Net

- The relationships in the network shown in Figure can be expressed in casual form of logic as shown :

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

Extended Semantic Networks for KR

- Entire **semantic net** can be coded using binary (two-argument) representation.

- Such representation is advantageous when additional information is added to the given system.

- **Example:**

  Jonh gave an apple to mike in the kitchen.

  location(E, kitchen)

Extended Semantic Networks for KR

- In FOL, predicate relation can have *n* arguments, where *n>=1.*

- **Example:**

  John gave an apple to mike.

- Can be represented in predicate logic by:
  give(john,mike,apple)

  Where,
  give represents the predicate relation
  john,mike and *apple* are the arguments.

Extended Semantic Networks for KR

Advantage of predicate logic representation compared to semantic net representation :

•In predicate logic representation - general prepositions can also be expressed, in addition to simple assertions.

Example:

John gave an apple to everyone he likes.

• Can be represented in predicate logic by the clause as:

give(john, X, apple) ← likes (john, X)

Here,

symbol *X* is a variable representing any individual.

arrow represents the logical connective *implied by.*

left side *of* ← contains *conclusion(s)*

right side contains *condition(s)*

Extended Semantic Networks for KR

Disadvantage of predicate logic representation:

•It is not convenient to add new information in an *n-ary* representation of predicate logic.

Example:

John gave an apple to mike.

give(john,mike,apple) -------- *3-ary* representation

- An additional information about the kitchen in the sentence:

  John gave an apple to mike in the kitchen

- Would have to replace *3-ary* representation with a new *4-ary* representation, such as:

  give(john,mike,apple,kitchen) -------- *4-ary* representation

Extended Semantic Networks for KR

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.

For example:

- The sentence *if john gives something he likes to a person, then he also likes that person* can be expressed in clausal representation in logic as

$$likes(john, X) \leftarrow give(john, X, Y), likes(john, Y)$$

- The sentence *every human is either male or female* is expressed by the following clause

$$male(X), female(X) \leftarrow human(X)$$

Extended Semantic Networks for KR

Disadvantage of conventional semantic network:

- Cannot express clausal form of logic.

Extended Semantic Network (ESNet):

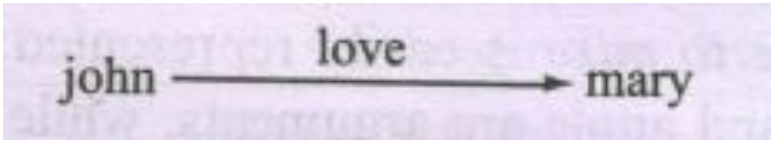
- To overcome these shortcomings, R Kowalski and his colleagues (1979) proposed ESNet.
- **ESNet combines the advantages of both logic and semantic networks**
  - ESNet can be interpreted as a variant syntax for the clausal form of logic- it has same expressive power as that of predicate logic.
  - Also incorporates the advantages of using binary relation as in semantic network rather than *n-ary* representation of predicate logic.
  - ESNet is a much powerful representation as compared to predicate logic and semantic network.

Extended Semantic Networks for KR

Representations in ESNet:

(i) 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 labelled as *love* with two end nodes representing *john* and *mary.*
  - The direction of the *arc (link*) indicates the order of arguments of the predicate symbol which labels the arc:

Extended Semantic Networks for KR

(ii) The clausual rule

male(X),female(X) <- human(X)

can be represented using binary representation as:
isa(X,male),isa(X,female) <- isa(X,human)

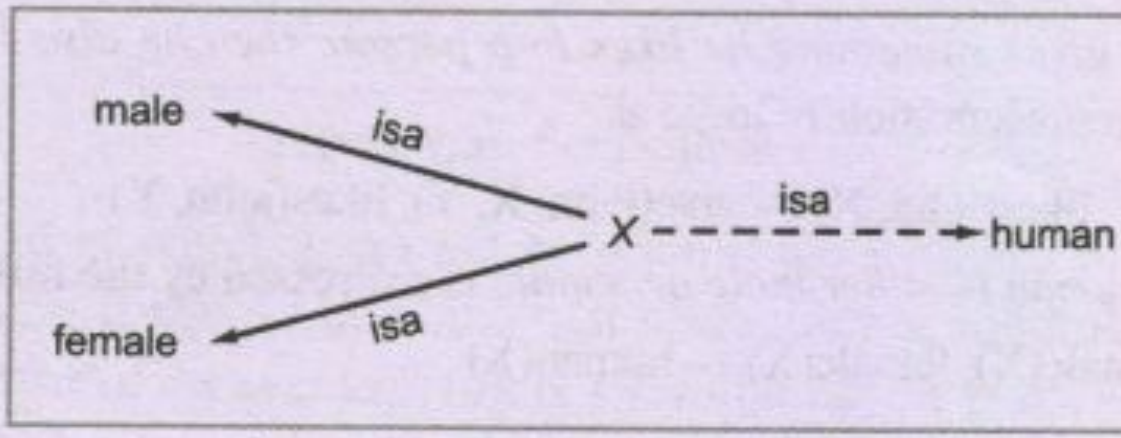


Figure 7.5 ESNet Representation

Inference Rules

• We have seen that any clause can be easily represented using ESNet which was not possible in ordinary Semantic Net.

•Now, we will discuss the inferencing rules associated with ESNet.

•Inference rules are embedded in the representation itself.

•Some of the inference rules are:

Inference Rules

(i) The representation of inference for every action of *giving*, there is an action of *taking* in clausal logic is :

action(f(x), take) <- action (X,give)

- The interpretation of this rule is that the event of *taking* action is a function of the event of *giving* action.
- In ESNet representation, functional terms, such as f(X), are represented by a single node.
- The representation of above statement in ESNet is :

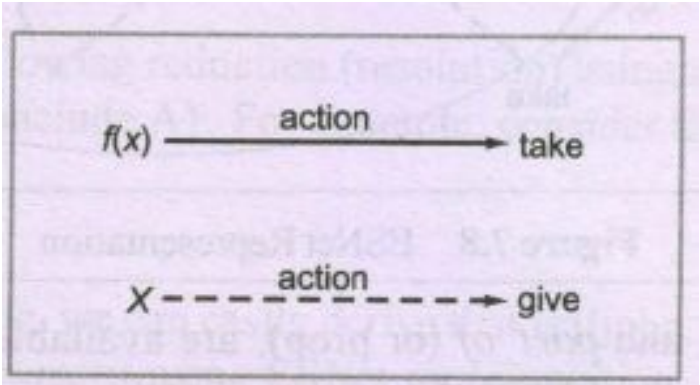


Figure 7.6 ESNet Representation

Inference Rules

(ii) The inference rule that an actor who performs a *taking* action is also the *recipient* of this action and can be easily represented in clausal logic ; ESNet as given below:

Here,

E is a variable representing an event of action of taking

recipient(E,X) <- action(E,take), actor(E,X)

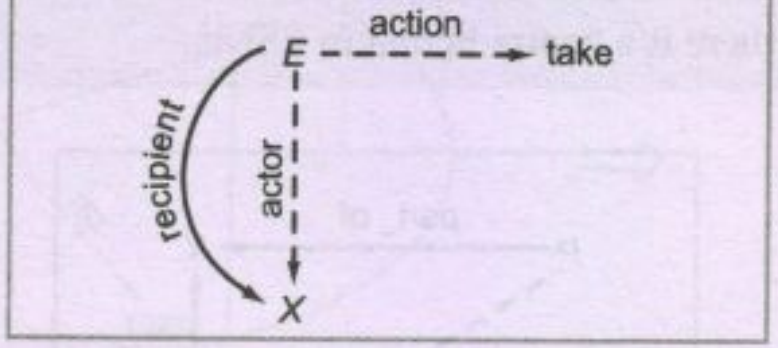


Figure 7.7 ESNet Representation

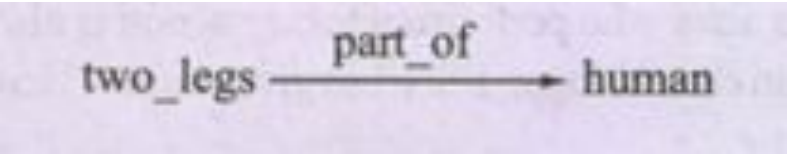
Inference Rules

(iii) The hierarchy links *isa, inst* and *part_of* (or prop), available in semantic networks and are also available as special cases in ESNet.

- It is important to note that *part-of* link has a hidden existential quantifier.

Example:

- The assertion *every human has two legs* could be represented using *part_of* link as follows:



•The interpretation of above representation is that *for every human, there exist two legs which are part of that human.*

Inference Rules

- The contradiction in ESNet can be represented as shown below:

P part_of *X is conclusion and P* part_of *Y* is condition

Where,

Y is linked with *X* via is*a* link

- *Such kind of representation is contradictory and hence there is a contradiction in ESNet.*

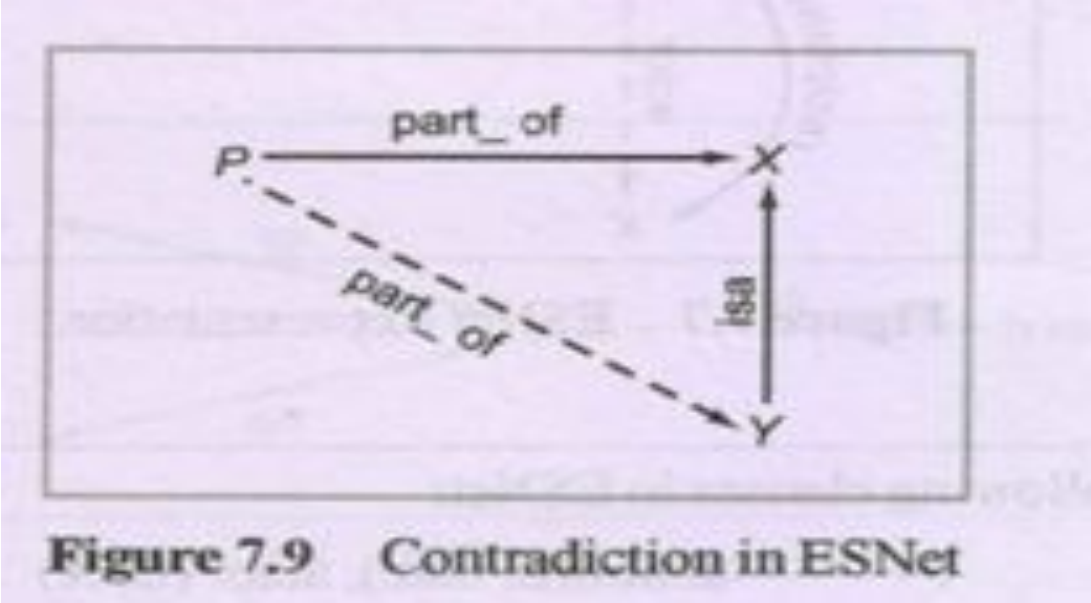


Figure 7.9 Contradiction in ESNet

Example:

7.1 Represent the following clauses in ESNet:

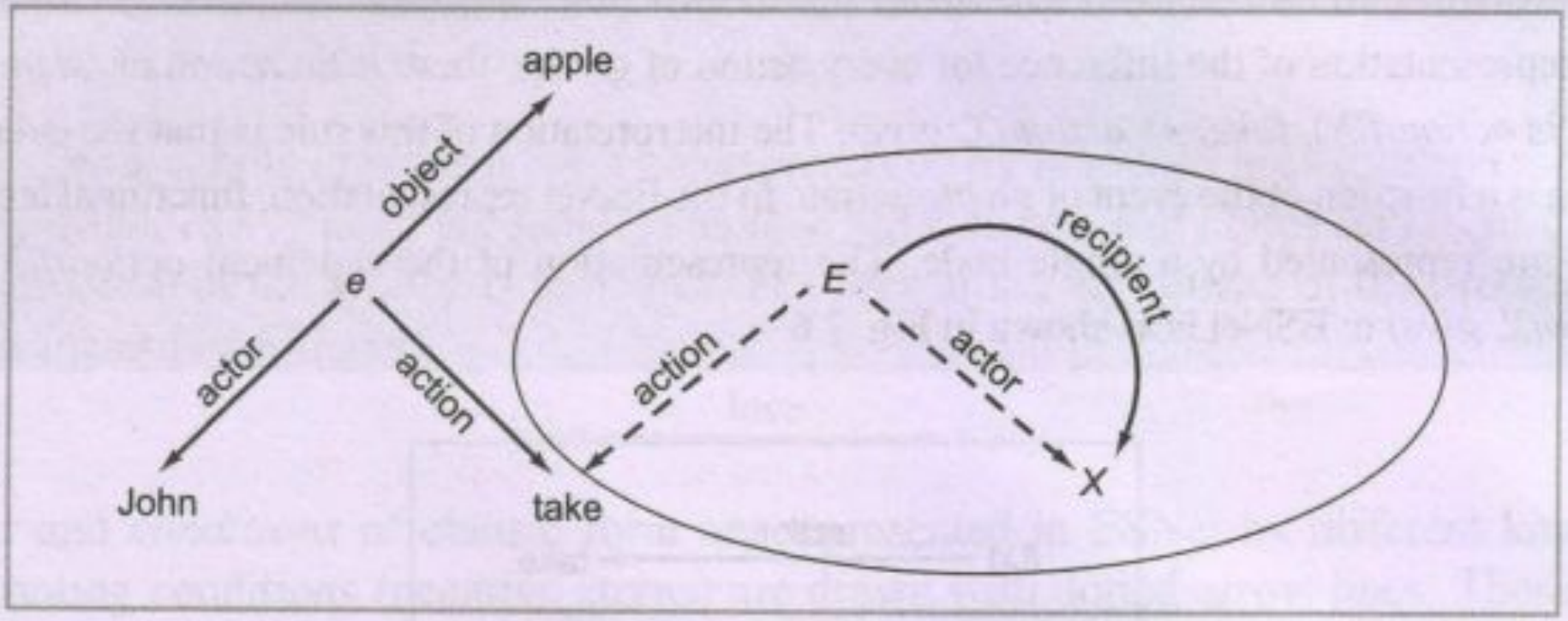
recipient(E, X) ← action(E, take), actor (E, X)

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 of the clauses is shown in Fig. 7.8. Here, the partition is shown to delimit the clause pictorially by enclosing the clausal rule in an ellipse.



Deduction in Extended Semantic Networks

- Inference rules discussed along with its semantics, form a part of ESNet.

- We know, in logic there are two types of inference mechanisms, namely:

  i. forward reasoning inference mechanism
  ii. backward reasoning inference mechanism

Forward and Backward Reasoning Inference Mechanisms

Forward Reasoning Inference Mechanism:

- Also called bottom-up approach.
- Forward Reasoning Inference Mechanism in clausal logic derives new assertion from the old ones.
  - i.e., in this mechanism, we start with the given assertions and derive new assertions uing clausal rule.

Backward Reasoning Inference Mechanism:

- Also called top-down approach.
- In backward Reasoning Inference Mechanism, we prove the query from the set of clauses using resolution refutation method in clausal logic

Both these inference rules are available in ESNet also.

Forward Reasoning Inference

- Given and ESNet, apply the following reduction (resolution) using modus ponen rule of logic
    - (A <- B) and B, then conclude A

Example:

- Consider the following set of clauses:

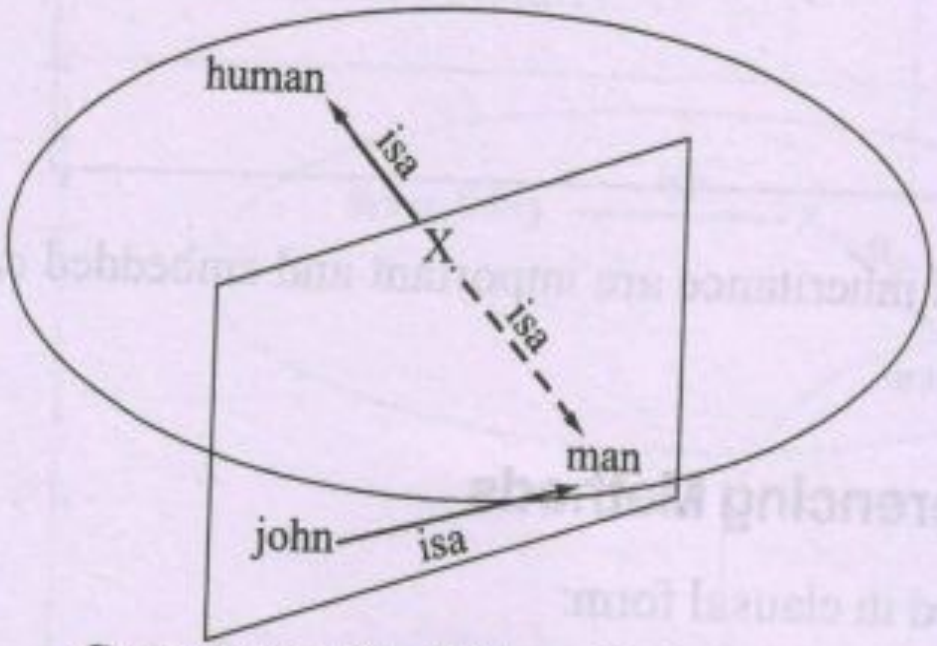
$$\text{isa}(X, \text{human}) \leftarrow \text{isa}(X, \text{man})$$
$$\text{isa}(\text{john}, \text{man}).$$

derive the new assertion *john is a human*

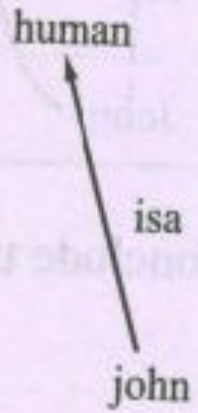
Forward Reasoning Inference

Table 7.4 Forward Reasoning Inference Mechanism

Given set of clauses	Inferencing
isa (X, human) ← isa (X, man) isa (john, man).	isa (john, human)



Contradiction after X is bound to john



Here the contradiction is enclosed in a rectangular box.

Backward Reasoning Inference

- In this mechanism, we can prove a conclusion or goal from a given ESNet by adding 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.

- **It is similar to proof by resolution refutation in clausal form of logic.**

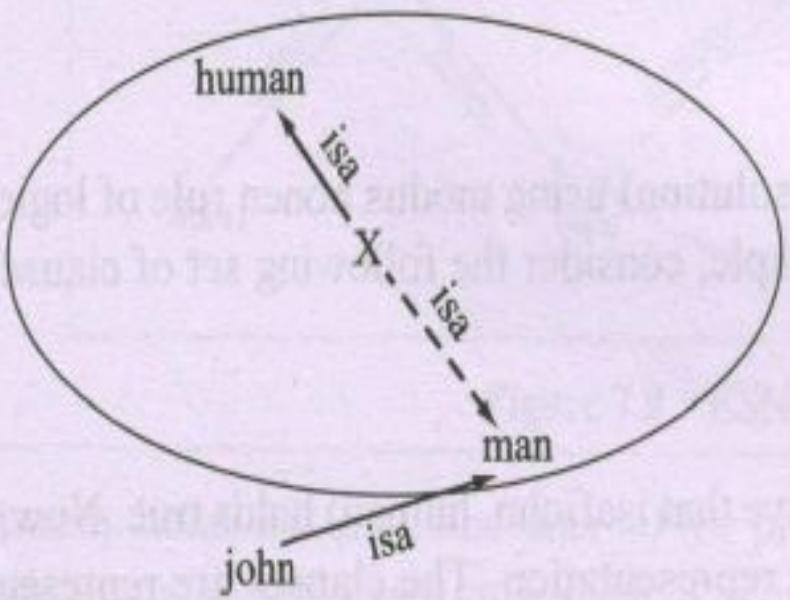
Example: Consider the set of clauses and prove *isa(john,human)*

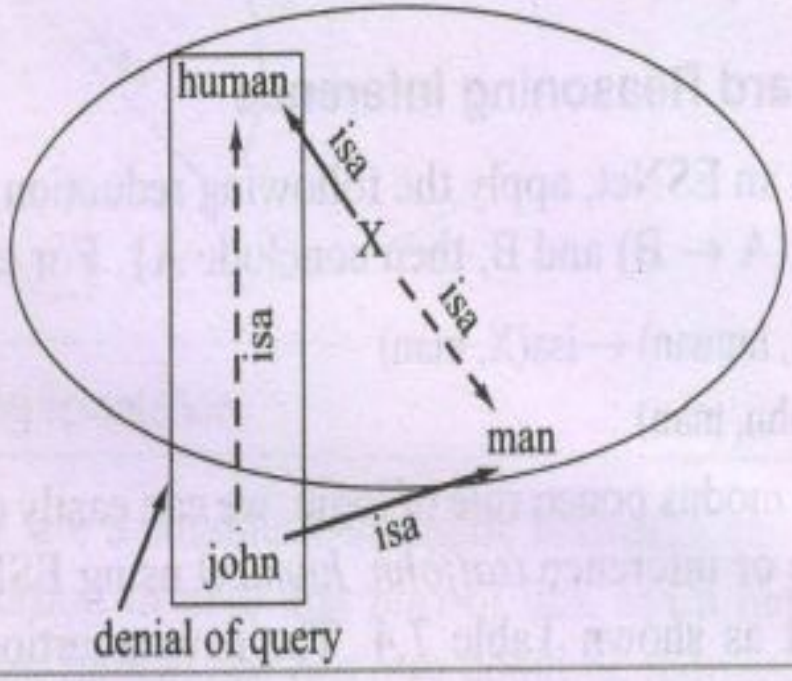
$$\text{isa}(X, \text{human}) \leftarrow \text{isa}(X, \text{man})$$
$$\text{isa}(\text{john}, \text{man}).$$

Backward Reasoning Inference

Table 7.5 Backward Reasoning Inference

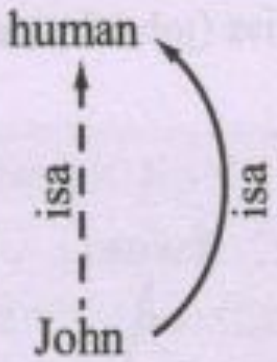
Given set of clauses	Prove conclusion
isa (X, human) ← isa (X, man) isa (john, man)	Query : isa (john, human)





Backward Reasoning Inference

Table 7.6 Reduction of ESNet

	X = john	Contradiction or empty network is generated. Hence, isa (john, human) is proved.

Examples of Illustrating Inferencing Methods

- Consider the following set of clauses:

```
isa(X, living_thing)  ←  isa(X, animate)
isa(X, animate)       ←  isa(X, human)
isa(X, human)         ←  isa(X, man)
isa(john, man)
```

Examples of Illustrating Inferencing Methods

Example:

Consider the set of clauses

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

Infer that *John is animate*

Examples of Illustrating Inferencing Methods

isa(X, living_thing) ← isa(X, animate)

isa(X, animate) ← isa(X, human)

isa(X, human) ← isa(X, man)

isa(john, man)

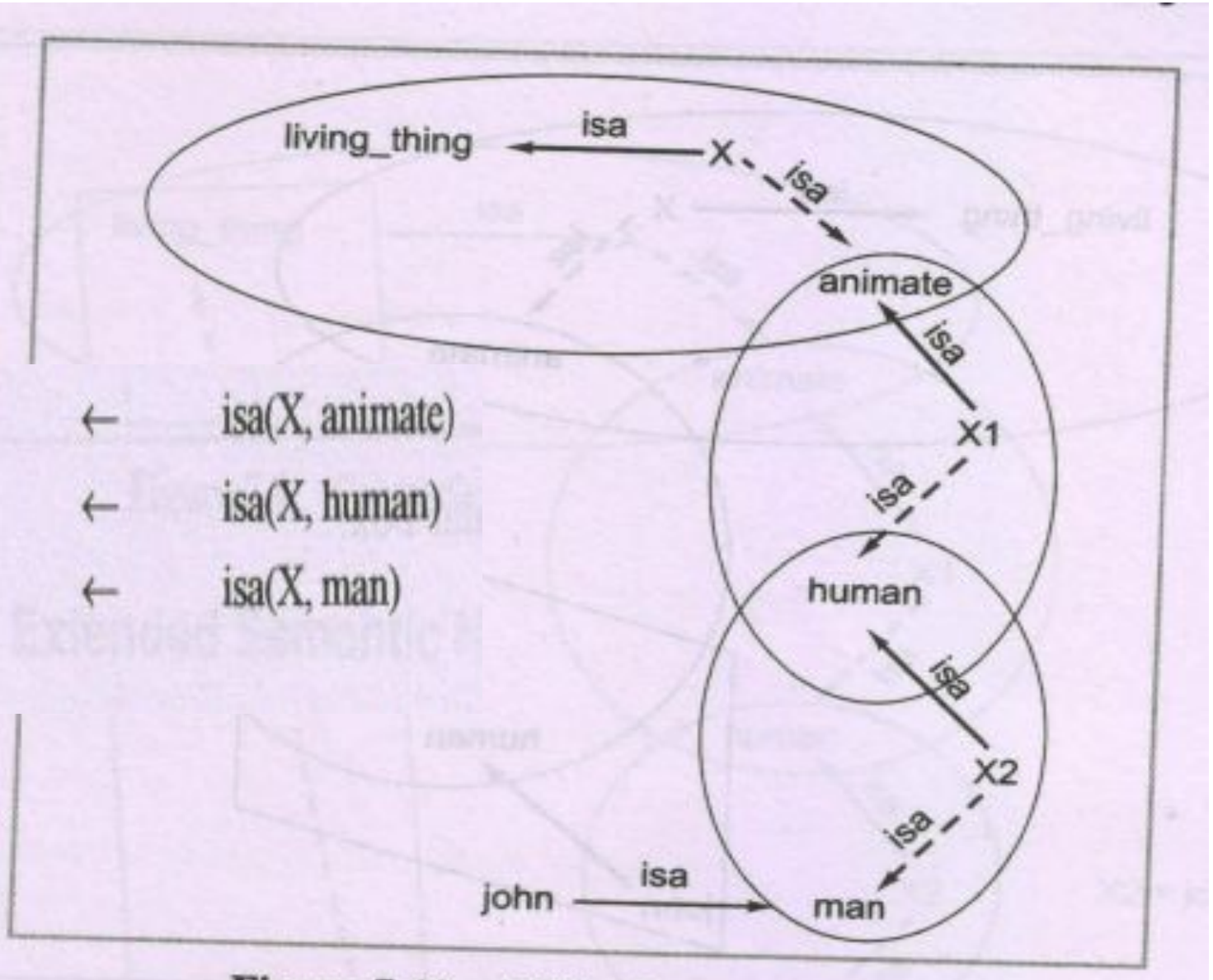
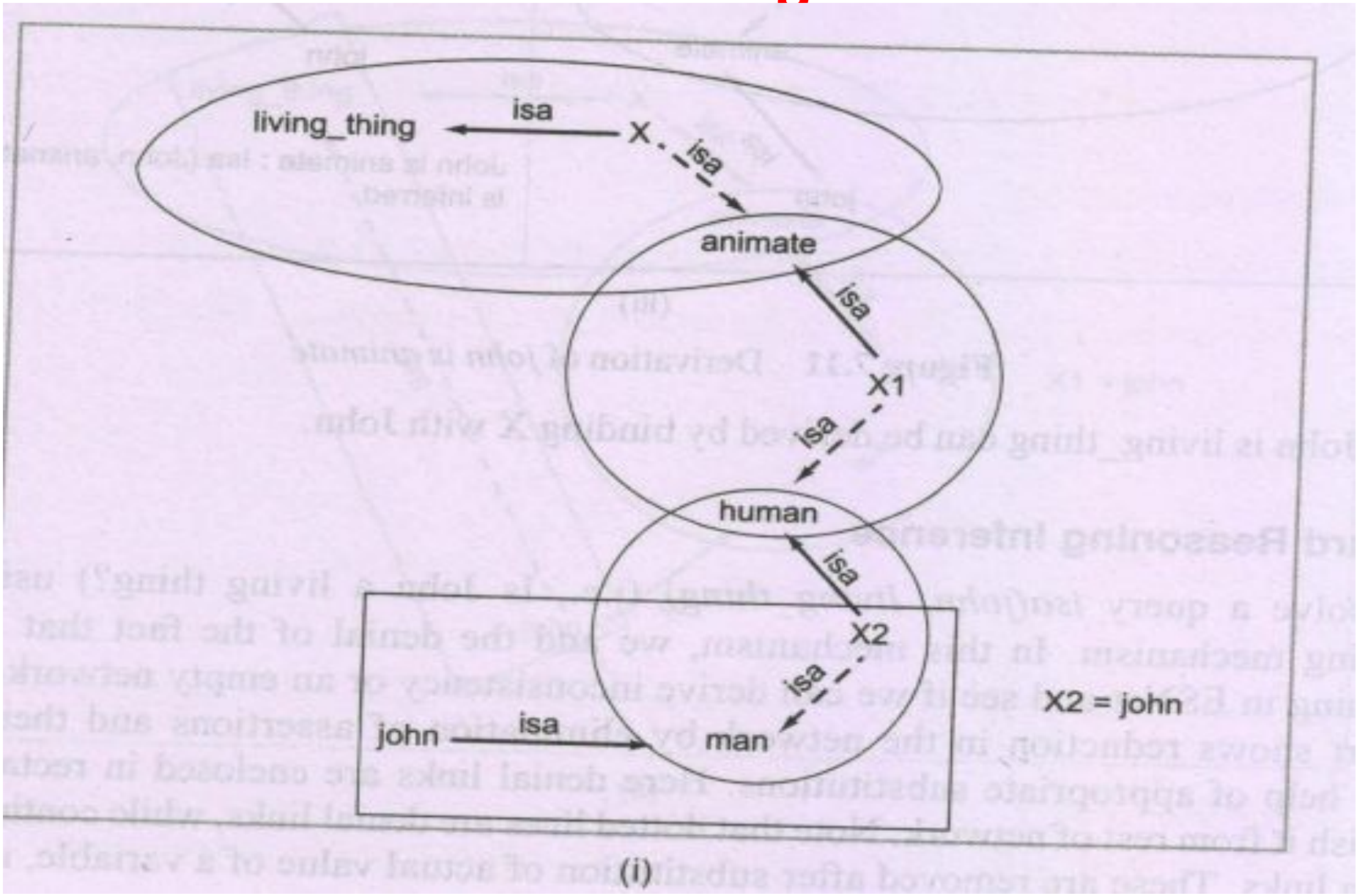
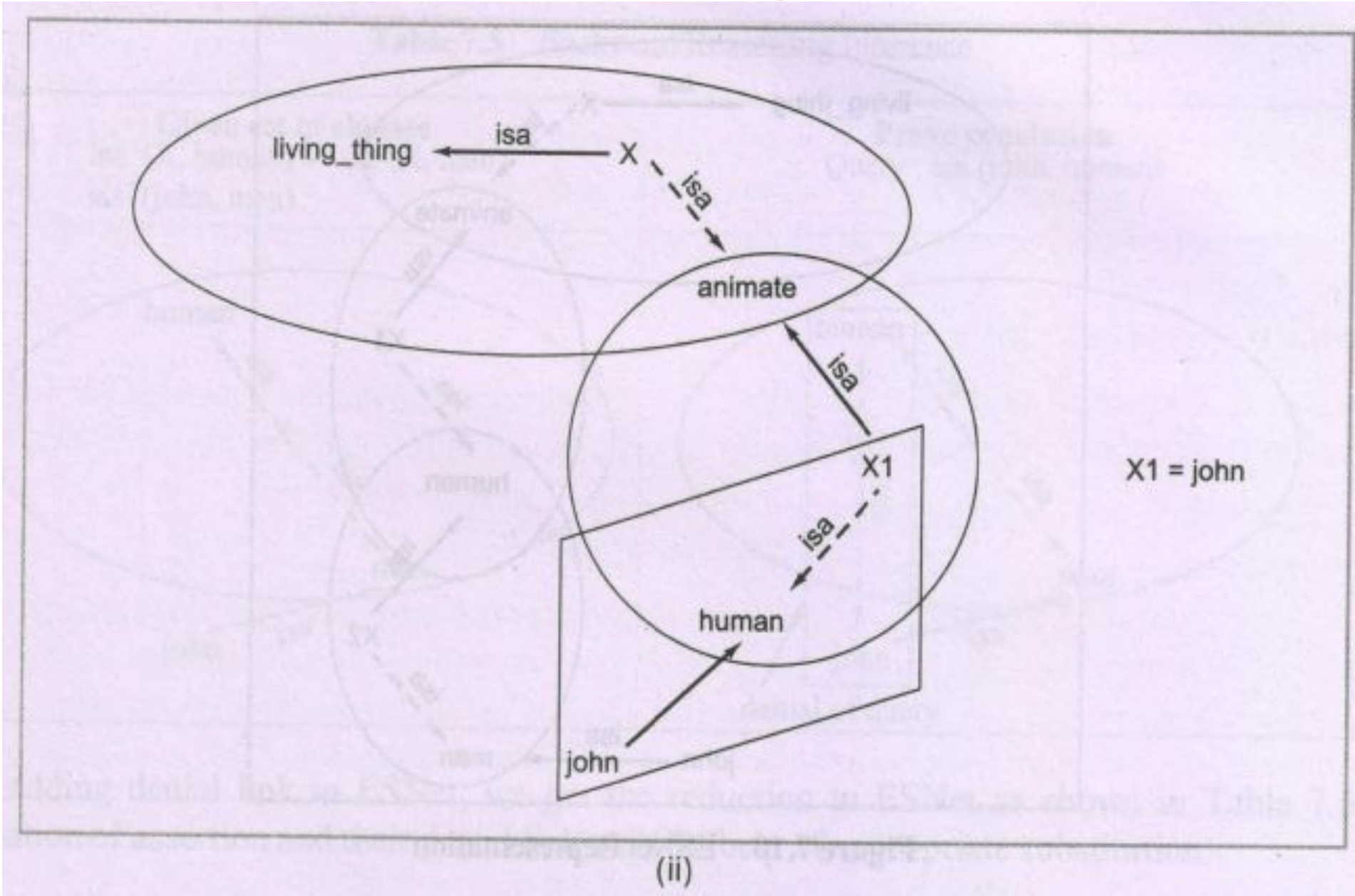


Figure 7.10 ESNet Representation

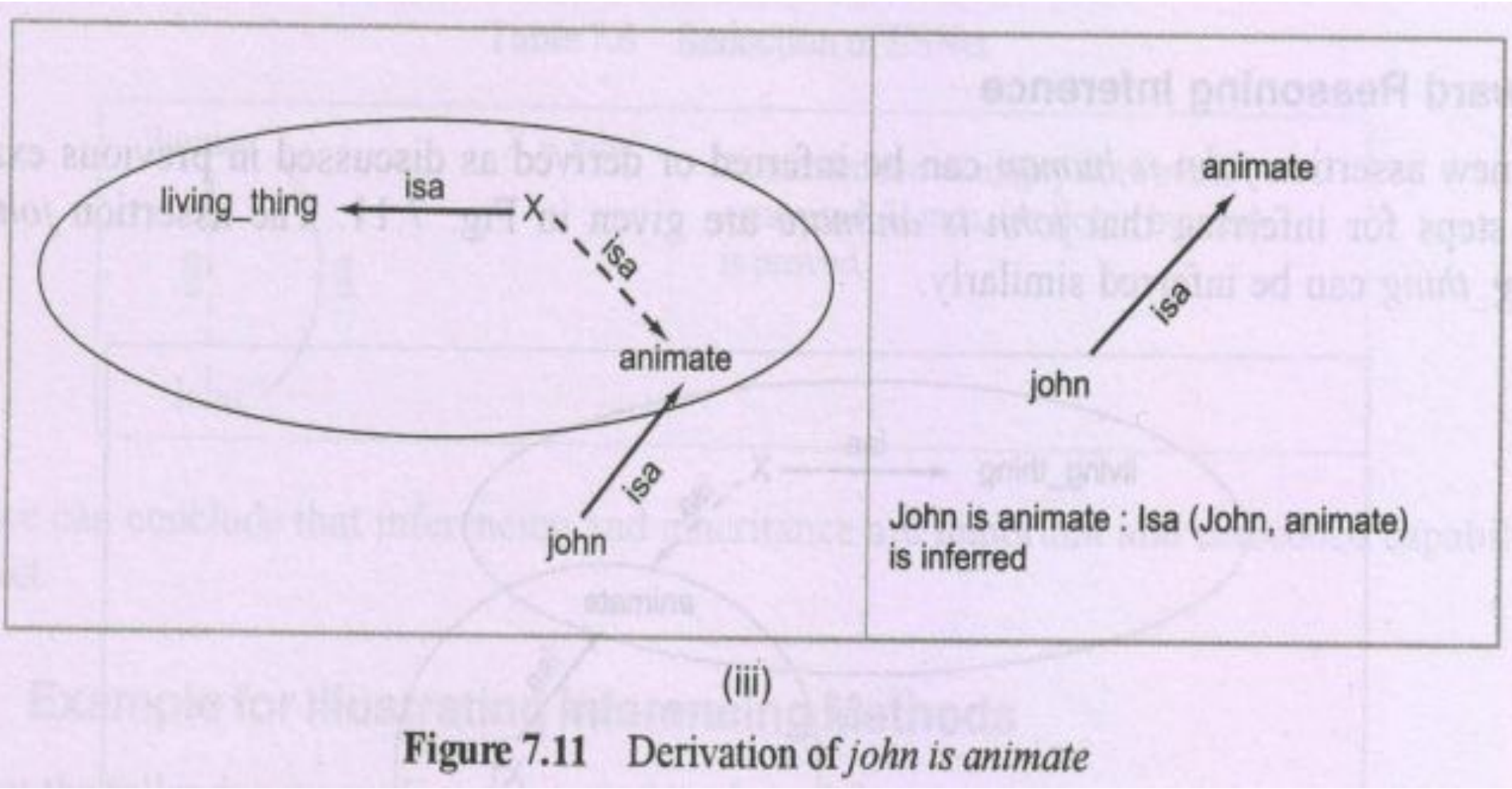
Forward Reasoning Inference



Forward Reasoning Inference



Forward Reasoning Inference



(iii)

Figure 7.11 Derivation of *john is animate*

- Futher in the same way *john is a living_thing* can be inferred similarly by binding *X* with *john*.

Backward Reasoning Inference

- Solve the query *isa(john,living_thing)* (i.e., Is john a living thing?)

- In this mechanism, add denial of fact that *John is not a living_thing* in ESNet and see if inconsistency or an empty network can be derived.

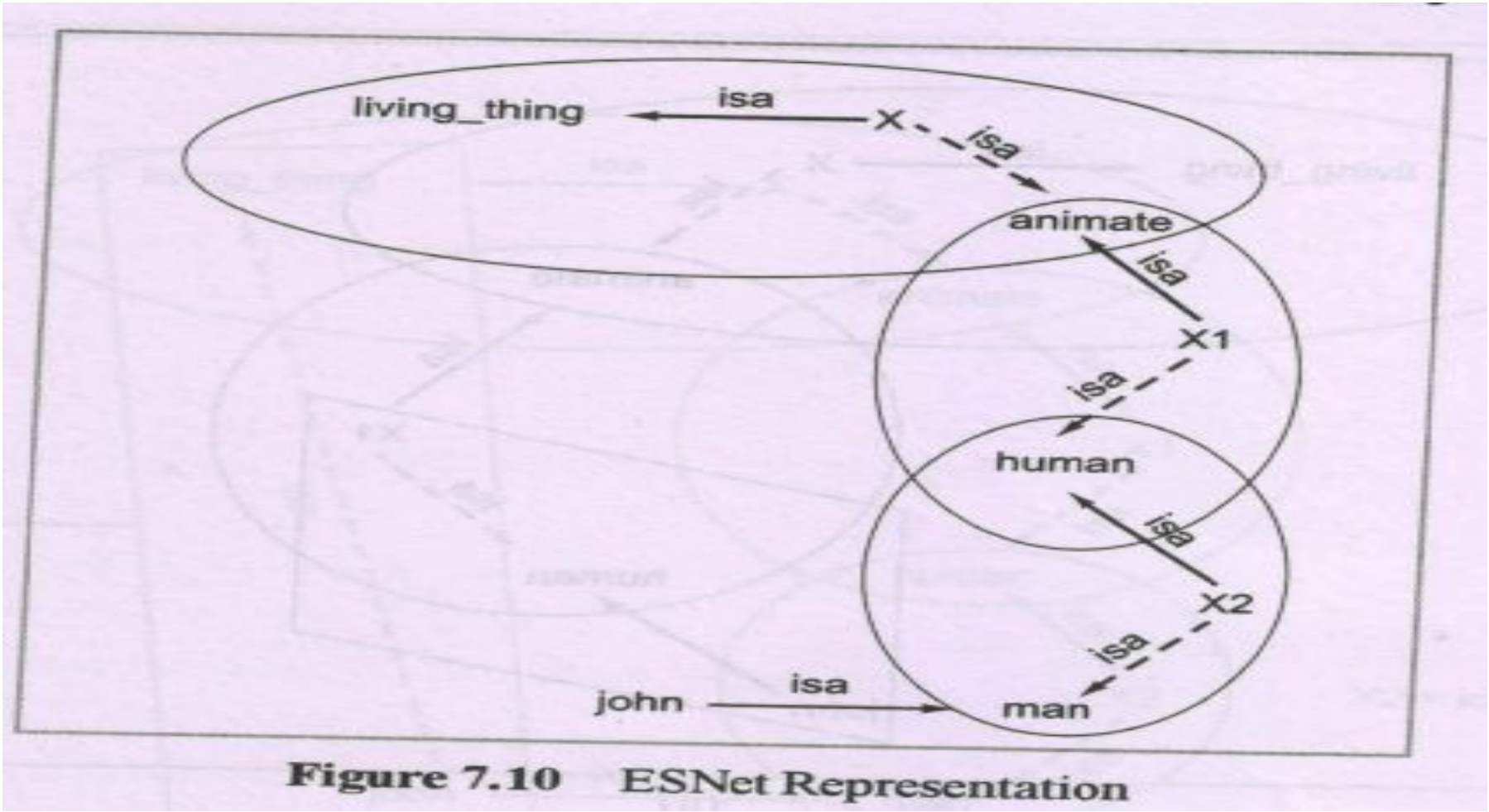
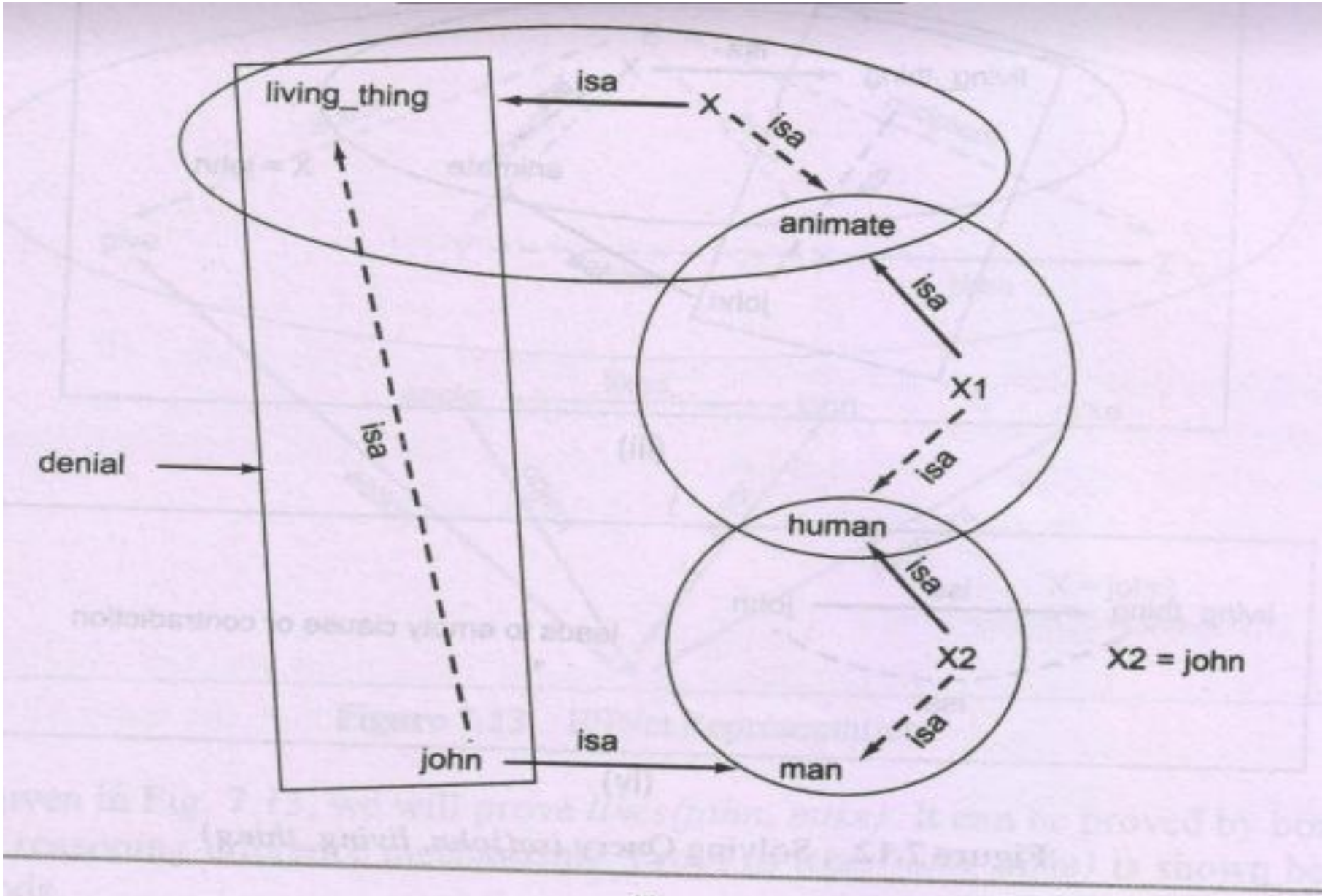


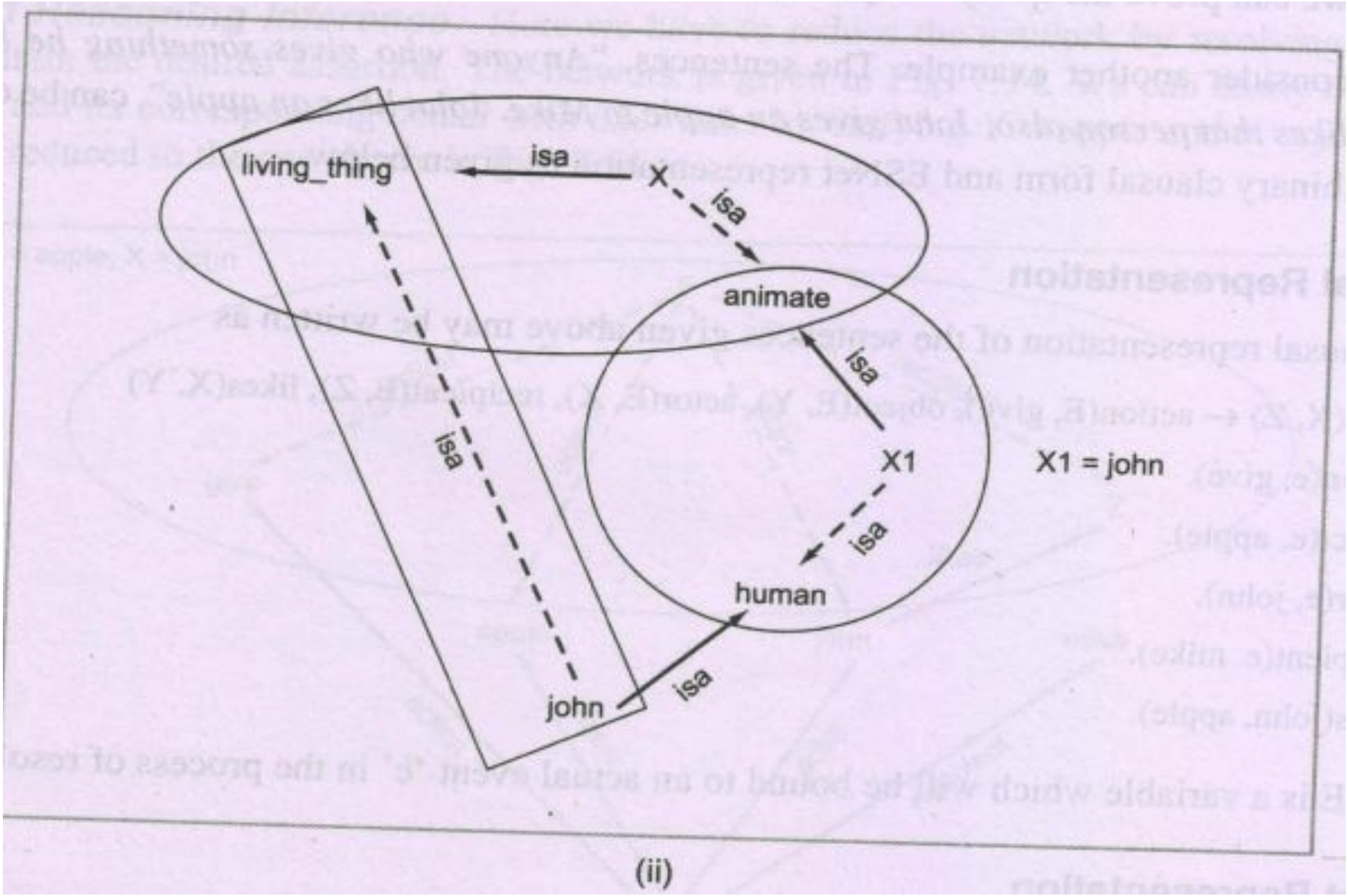
Figure 7.10 ESNet Representation

Backward Reasoning Inference

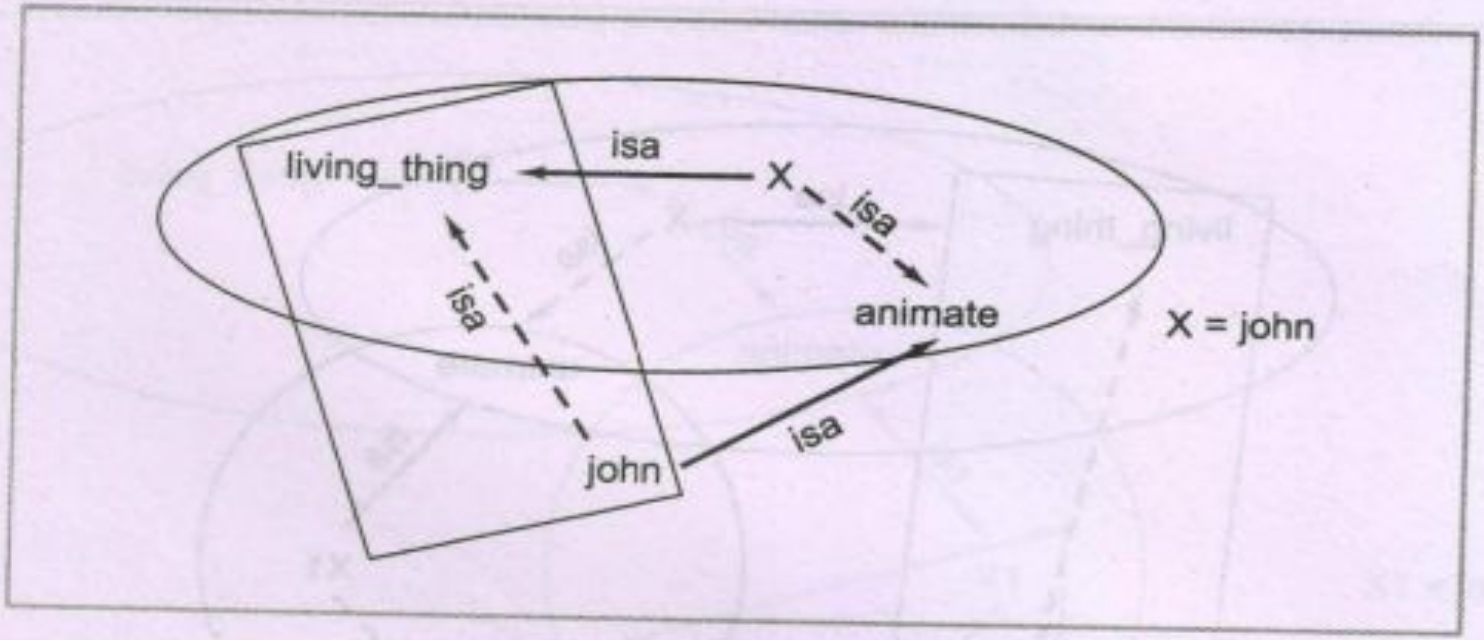


(i)

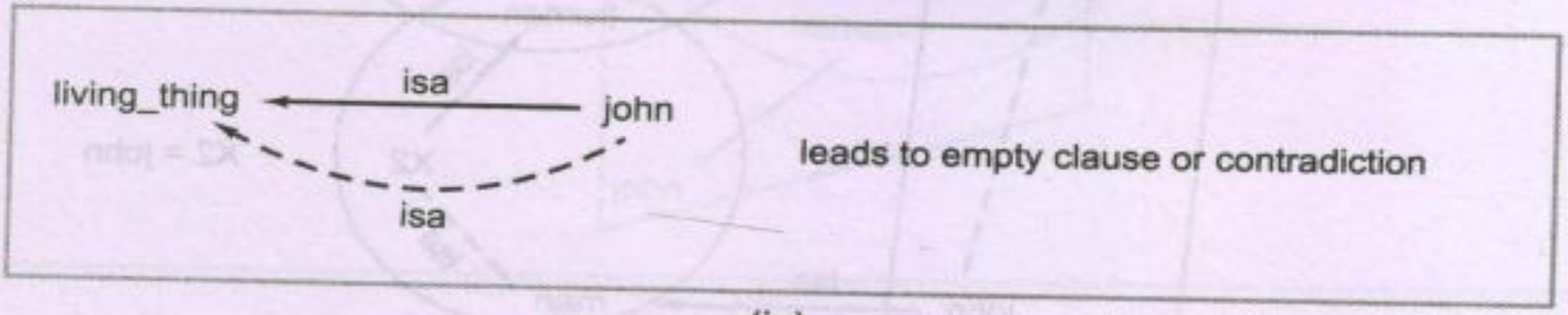
Backward Reasoning Inference



Backward Reasoning Inference



(iii)



(iv)

Figure 7.12 Solving Query *isa(john, living_thing)*

Backward Reasoning Inference

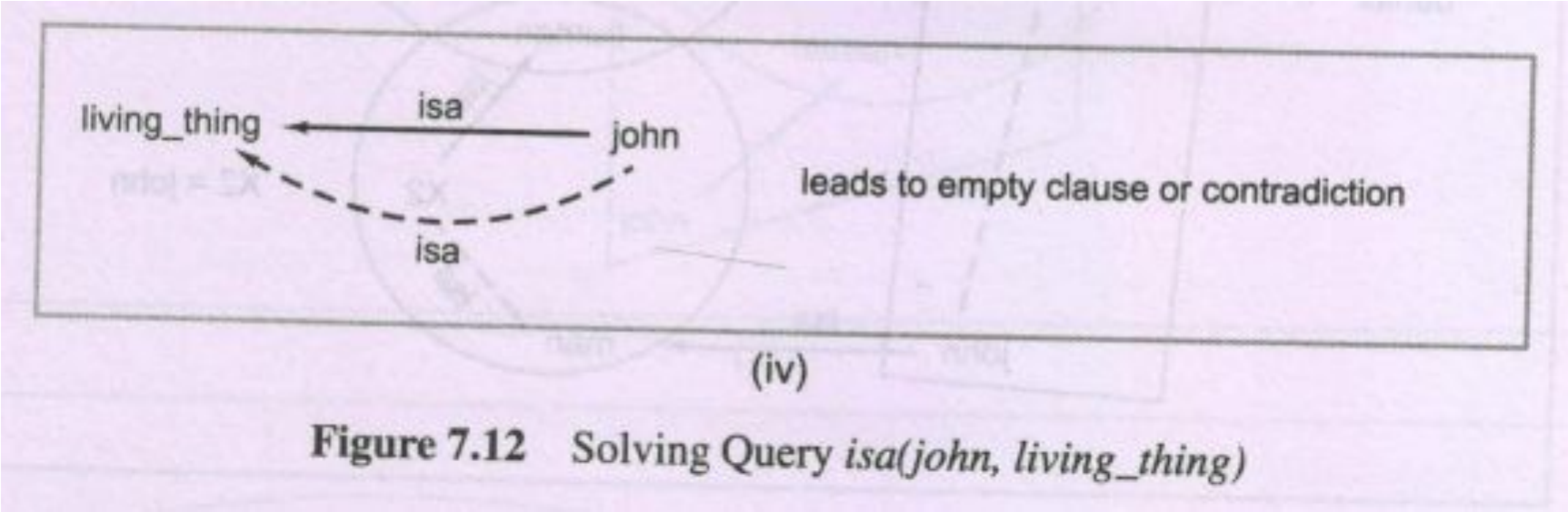


Figure 7.12 Solving Query *isa(john, living_thing)*

- Hence, we can prove the query that john is an animate.

Clausal Representation

- **<u>Let us consider another example.</u>**

- Sentence:

  " anyone who gives something he likes to a person likes that person also.
  John gives an apple to Mike. John like an apple"

- Clausal representation of the sentences given above may be written as:

likes(X, Z) ← 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 which will be bound to an actual event 'e' in the process of resolution.

ESNet Representation

- The ESNet for the clauses described above is:

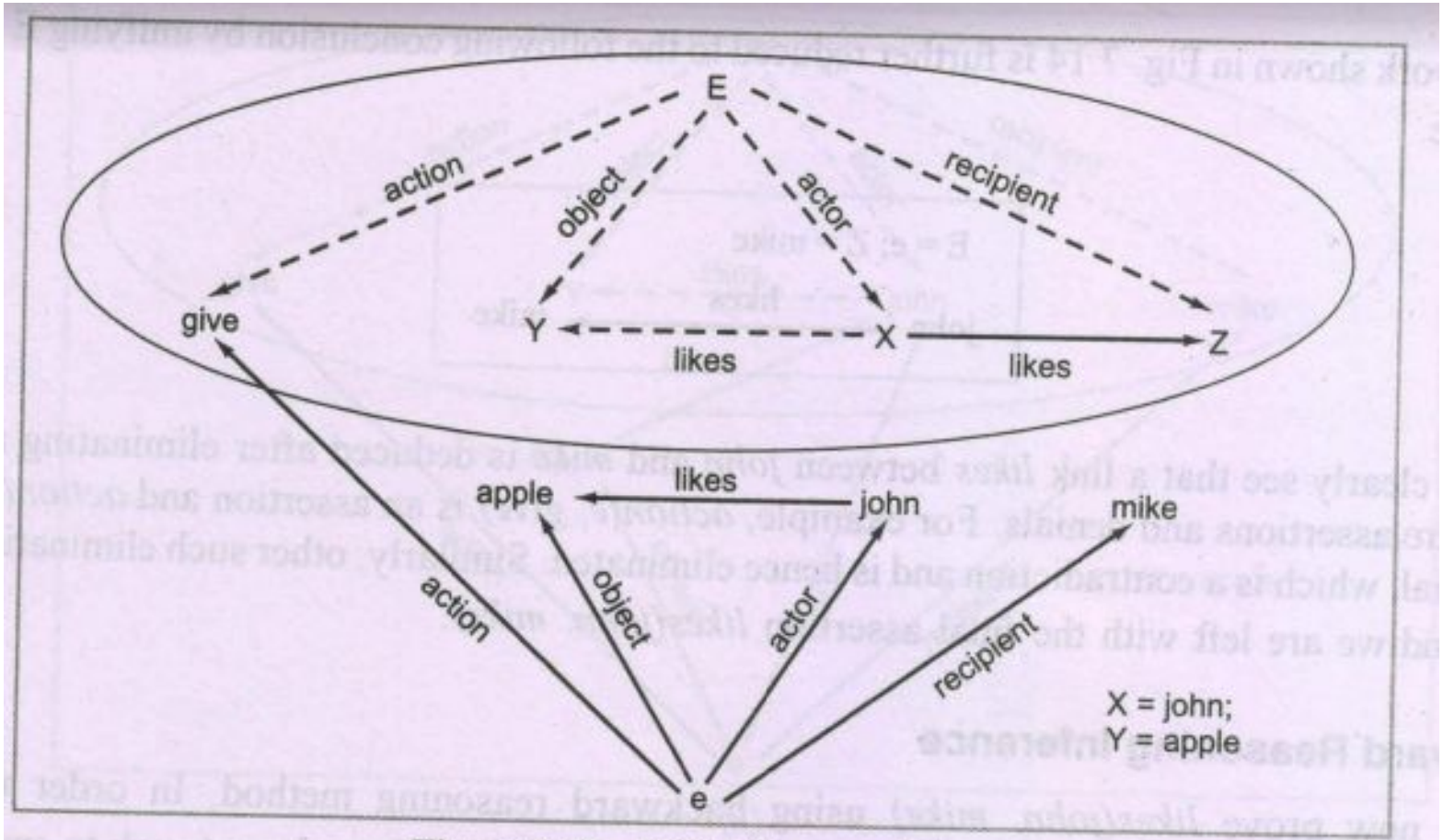


Figure 7.13 ESNet Representation

ESNet Representation

- From the ESNet give, we will prove:

  likes(john,mike)

- It can be proved by both

  i. forward reasoning inference mechanism
  ii. backward reasoning inference mechanism

(i) Forward Reasoning Inference

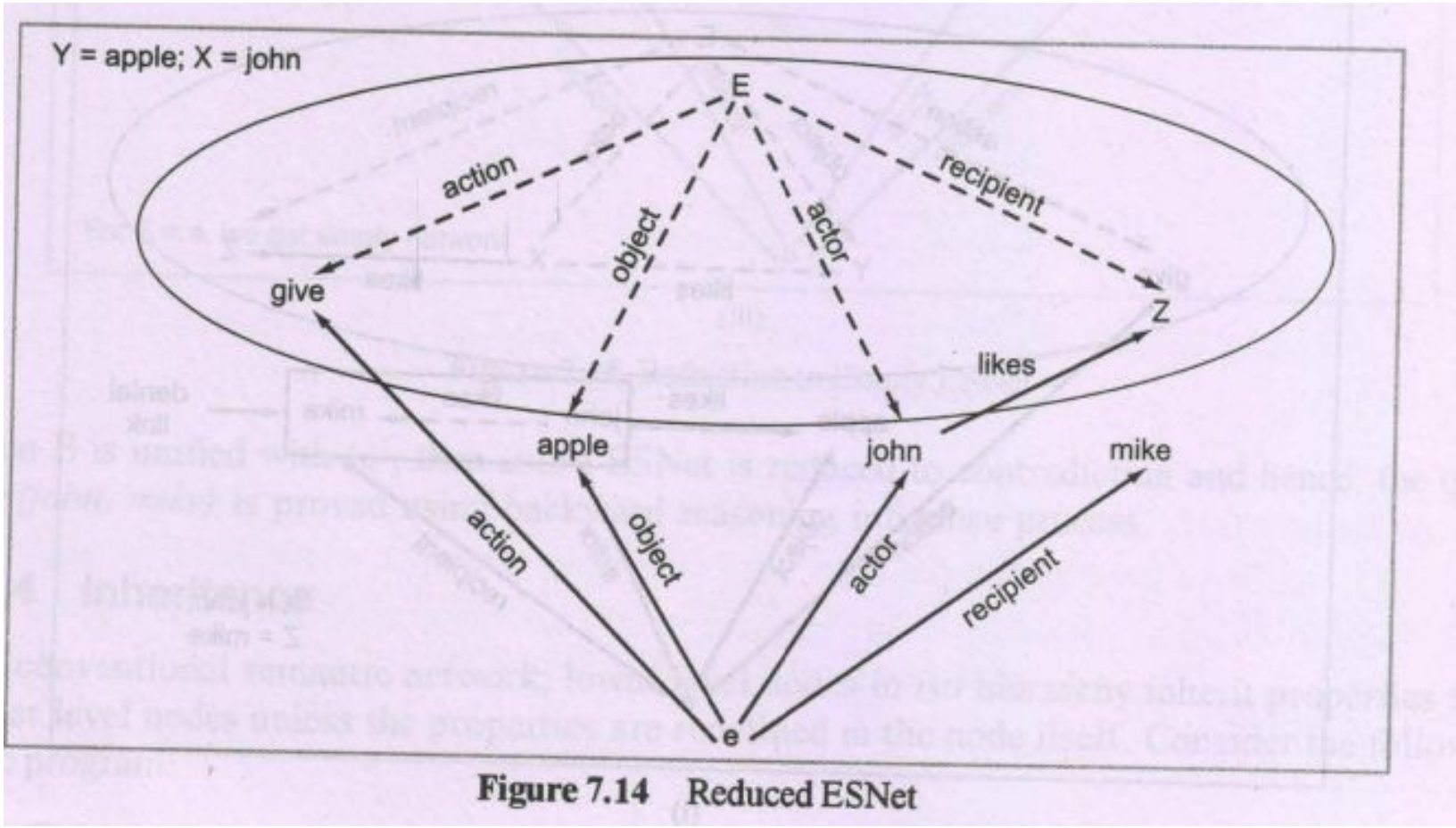
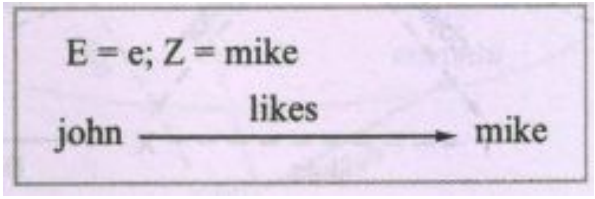
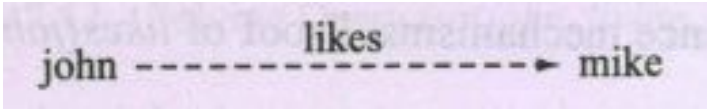
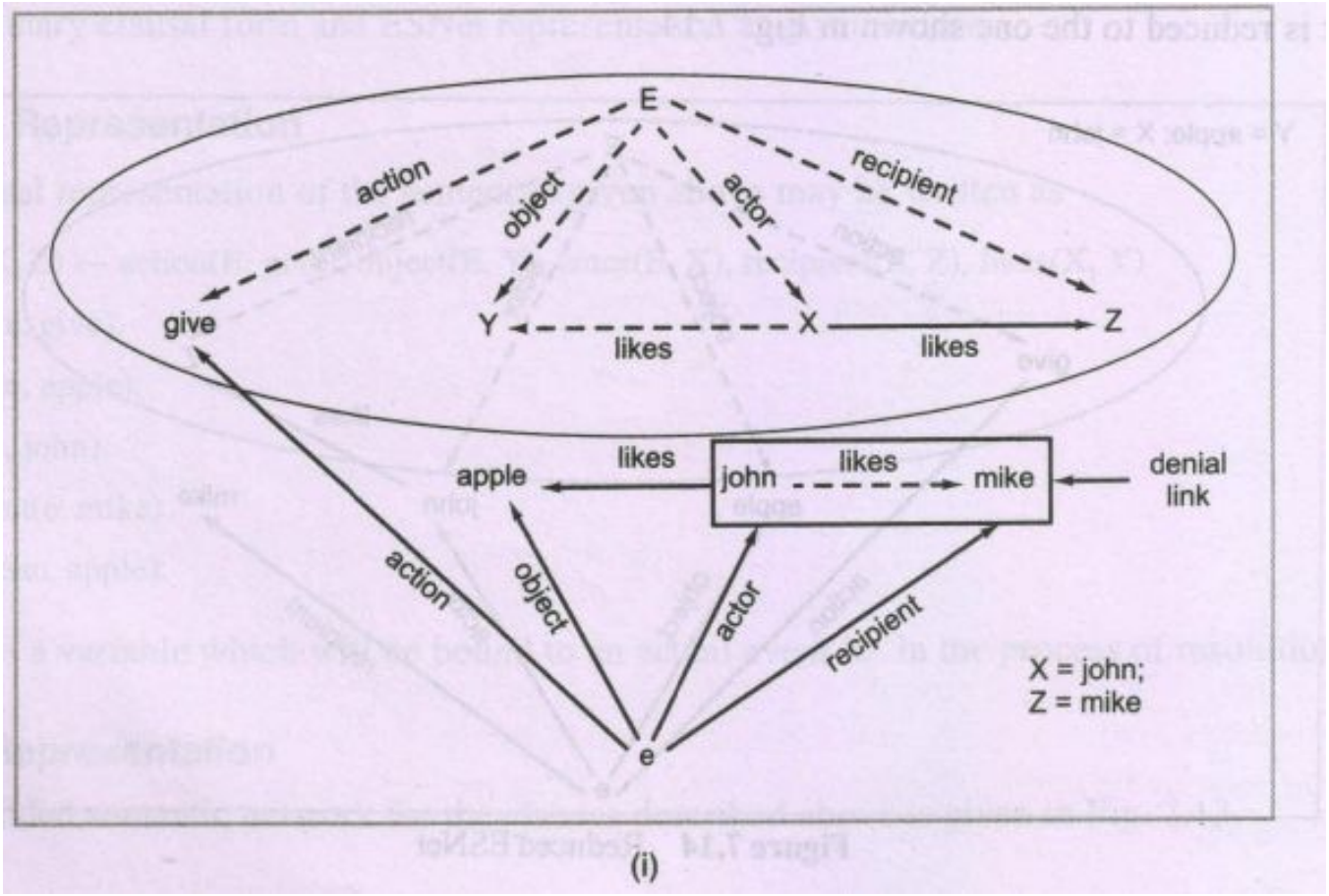


Figure 7.14 Reduced ESNet

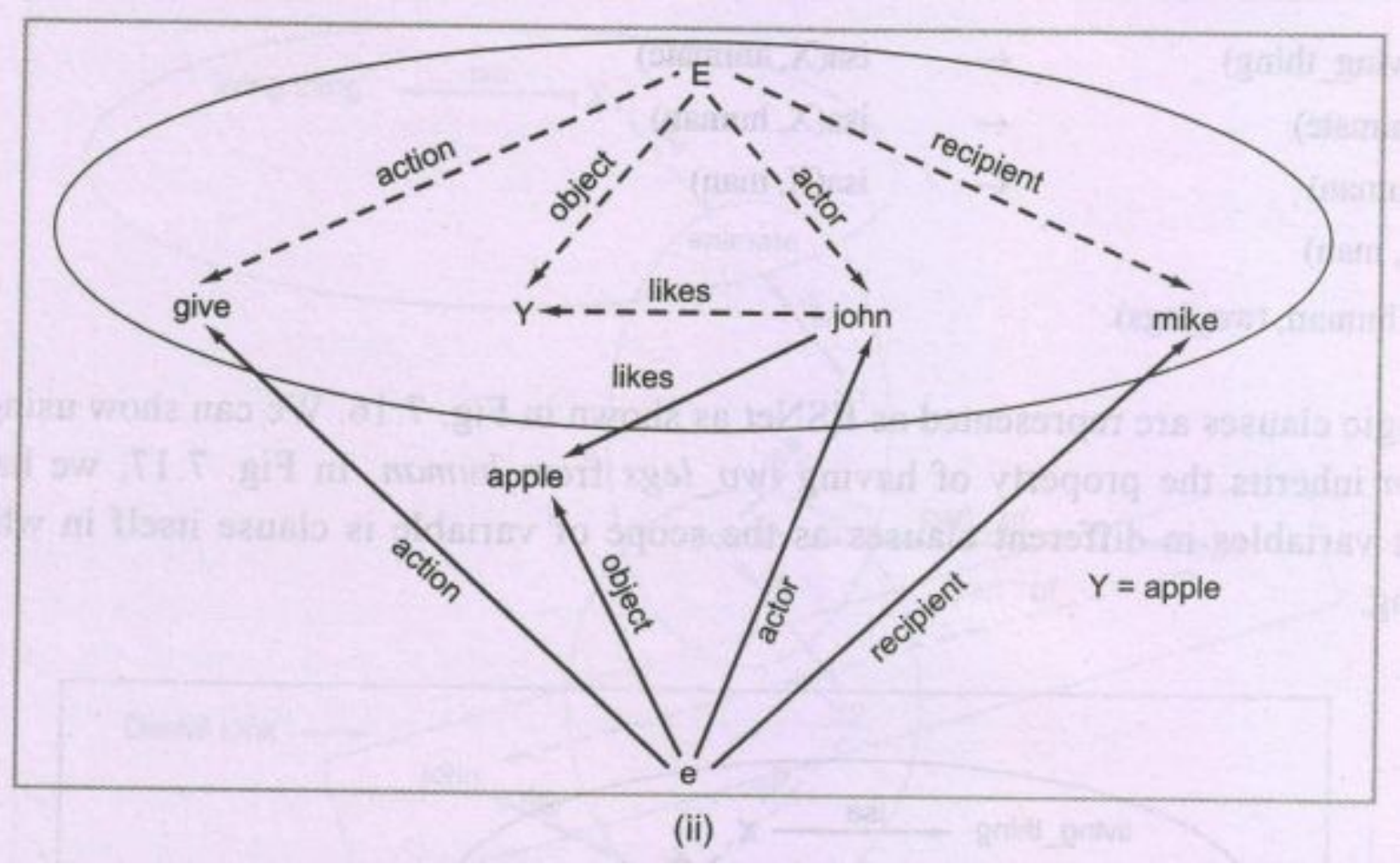


(ii) Backward Reasoning Inference

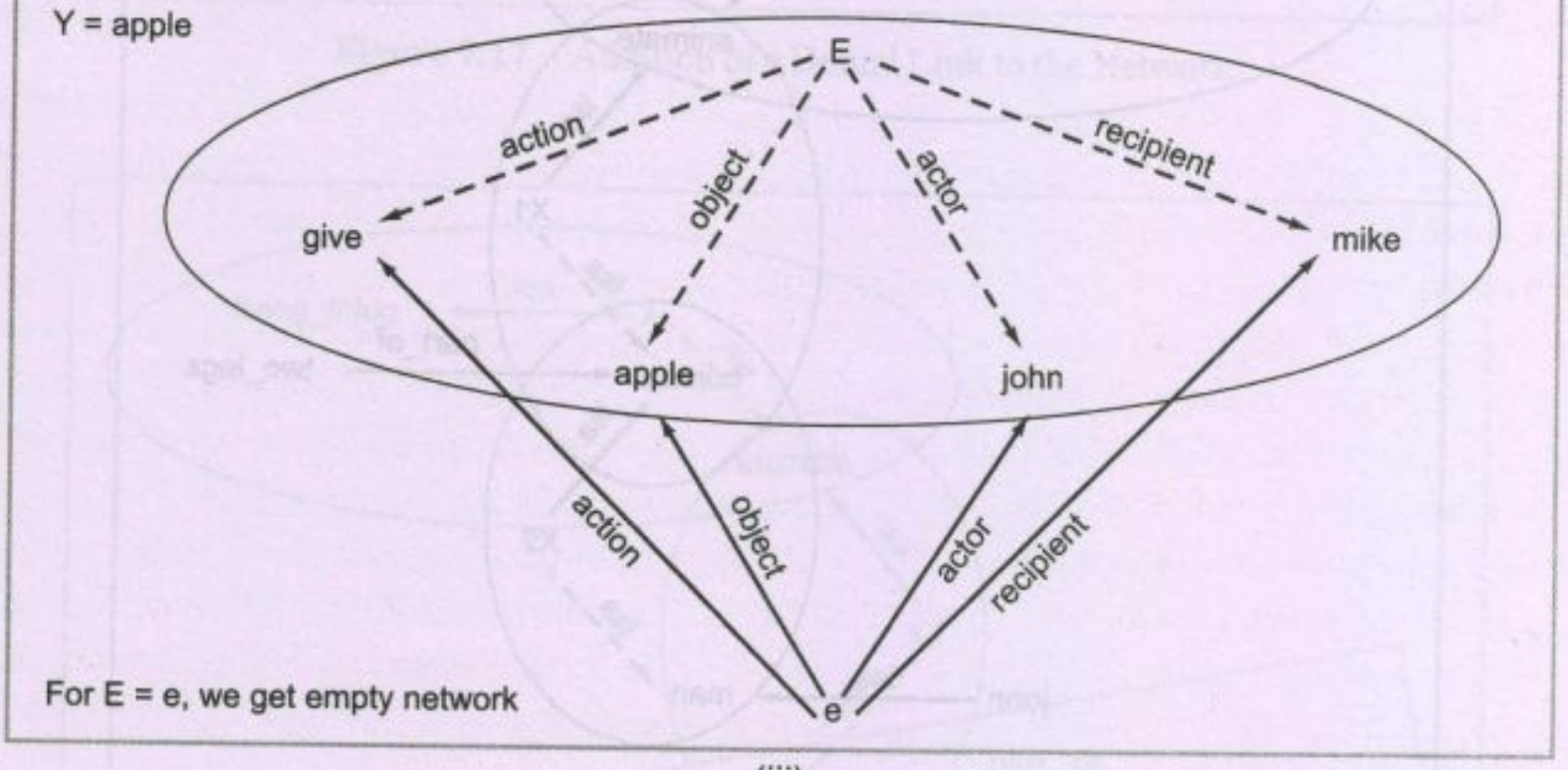




(ii) Backward Reasoning Inference



(ii) Backward Reasoning Inference



(iii)

Figure 7.15 Reduction to Empty ESNet

Inheritance

- In conventional Semantic network, lower level nodes in i*sa* hierarchy inherits properties from the higher level nodes unless the properties are redefined in the node itself.

Example:

Consider the following logic program:

```
isa(X, living_thing)  ←  isa(X, animate)
isa(X, animate)       ←  isa(X, human)
isa(X, human)         ←  isa(X, man)
isa(john, man)
part_of(human, two_legs).
```

- The above logic clauses can be represented in ESNet as shown.
- From the ESNet, it can be observed that *john* inherits the property of having *two_legs* from human.
- ESNet is represented by using different variables (X,X1,X2) in different clauses as the scope of variable is in clause itself in which it is appearing.

Inheritance

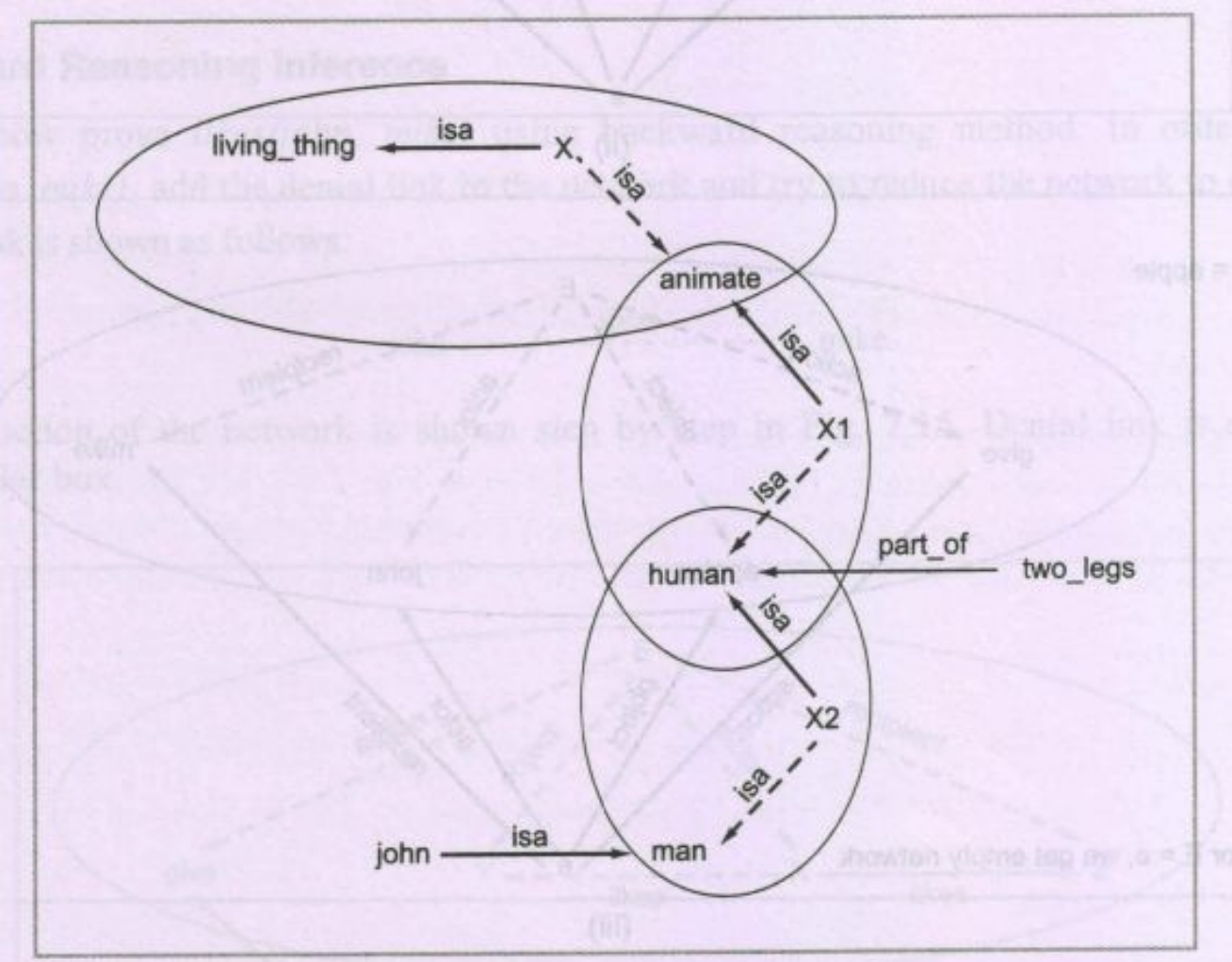


Figure 7.16 ESNet for Inheritance

Inheritance

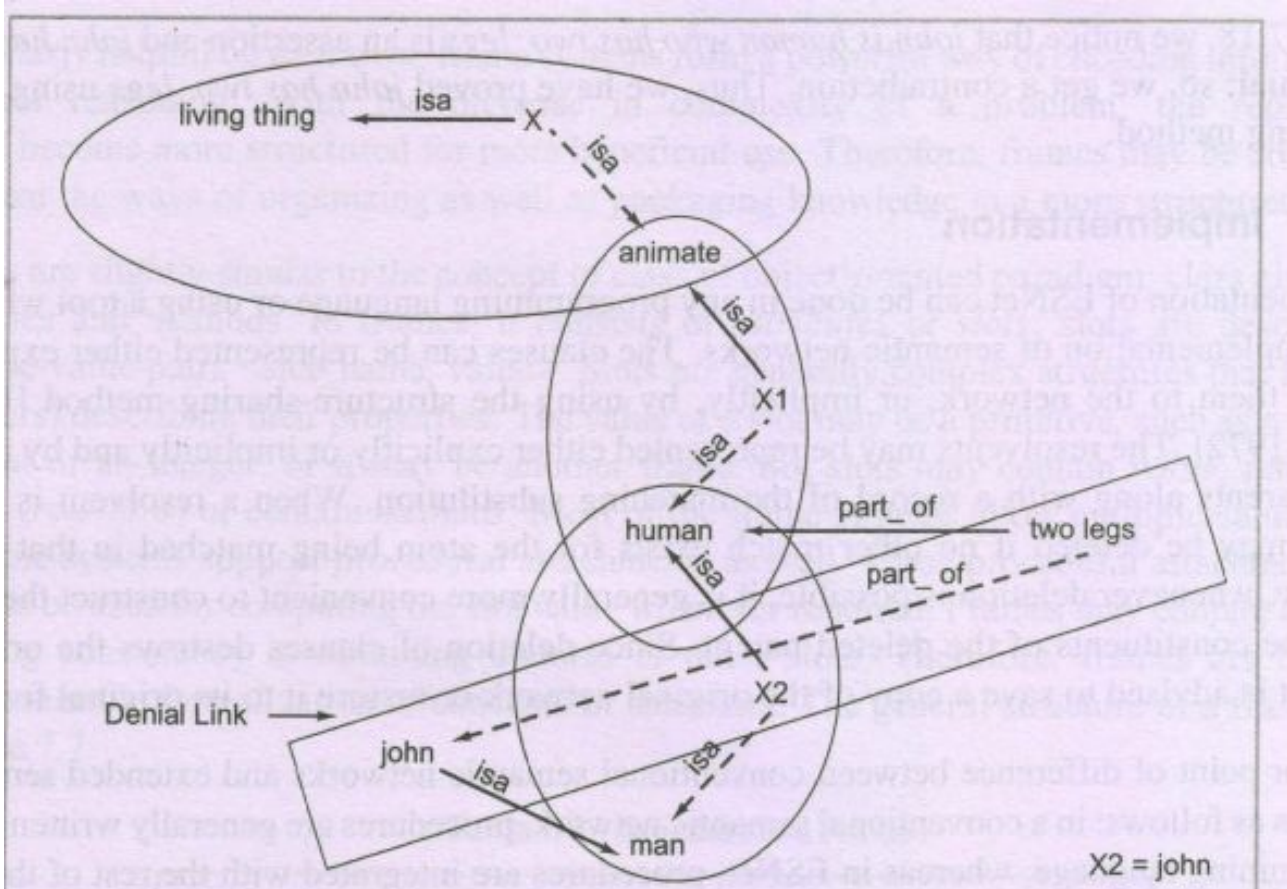


Figure 7.17 Addition of a Denial Link to the Network

Inheritance

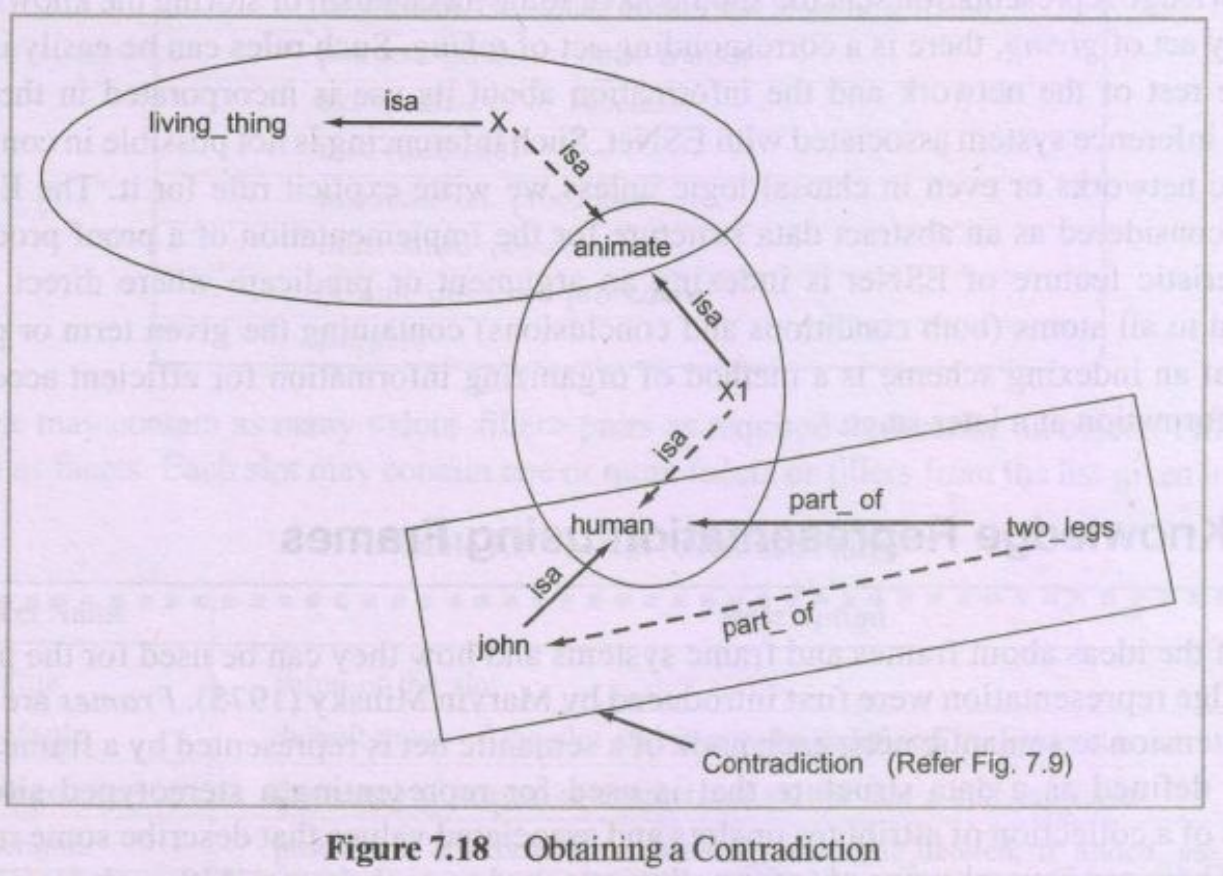


Figure 7.18 Obtaining a Contradiction

- It can be noticed that *john is human who has two_legs* is an assertion and *john has two_legs* is a denial, we get a contradiction.

- Thus, it has proved *john has two_legs* using backward reasoning method.

Implementation

- 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.
- The resolvents may be represented either explicitly or implicitly and by pointers to their parents along with a record of the matching substitution.
- When a resolvent is created, a parent may be deleted if no other match exists for the atom being matched in that parent.
- Major difference between conventional semantic network and ESNet:
    - In a conventional semantic network, procedures are generally written in the host programming language.
    - In ESNet, procedures are integrated with the rest of the database and are executed by the same general-purpose mechanism which perform inference in the network.

Example: *John gives an apple to mike* and if we wish to ask *who takes an apple*?

Knowledge Representation Using Frames

- Frames are the extension of semantic nets.
- Each node of a semantic net can be represented by a frame.
- Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations.
- A frame is a record like structure consists of a collection of attributes (or slots) and its associated values (slot values) to describe an entity in the real-world.
  - i.e., It consists of a collection of slots and slot values.
- There are several types of information attached to each frame.

Knowledge Representation Using Frames

- Frames are slightly similar to the concept of class of object-oriented paradigm.
- Slots consists of attributes or slots (~class contains attributes and methods).
- These slots may be of any type and sizes.
- Slots are described with attribute-value pairs <slot_name,value>.
  - names and values of slots are called facet.
- Facets are features (or properties) of frames which enables to put constraints on the frames.

Knowledge Representation Using Frames

- A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of values.
- A frame is also known as **slot-filter knowledge representation** in AI.
- The value of slot may be a primitive, such as a text string, constant or an integer or it may be any other frame.
- So, slots may contain value, referring to other frames (relations) or certain methods.
- Most of the frame systems allow multiple values for slots.

Knowledge Representation Using Frames

- The general structure of a frame is:

Table 7.7 Structure of a Frame

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

Knowledge Representation Using Frames

- A frame may contain many <**slot-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 below given list:

Table 7.8 List of Facets in a Frame

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

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, then descendent frames cannot modify that value; this value remains unchanged for all subclasses and instances of that class.
- The related frames are linked together into *frame systems* and are organized into hierarchies or network of frames.
- Each frame in the network is either a class frame or an instance frame.

Knowledge Representation Using Frames

Example:

A hospital frame has been defined below:

Table 7.9 Hospital Frame

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

Knowledge Representation Using Frames

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

- **Ako:** This link connects two class frames, one of which is a kind of the other class, e.g., 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. It gives a sub-typing hierarchy where all instances of class frame are instances of super class frames. For example, all child hospitals are hospitals but all hospitals may not be child hospitals. With the help of this link, knowledge representation becomes more structured and memory efficient.
- **Inst:** This link connects a particular instance frame to a class frame, e.g., 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, e.g., *ward* is Part_of the class *hospital*.

- Other required information may be made available using slot-value pair concept.

Frame Descriptions

Example:

- Network of frame system for hospitals can be described as:

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)

Frame Descriptions

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)

Frame Descriptions

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: (....)	

Frame Descriptions

X_Ray Frame

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

Orthopaedic Ward Frame

F_name:	orthopeadic
Ako :	ward (Ward Frame)

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:	Wards Frame Inst
Doctors:	Doctors Frame Inst

Frame Instance Descriptions

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

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])

Frame Instance Descriptions

Child Hospital Frame Instance

F_name:	kalawati
Inst:	Child_hospital (Child Hospital Frame)
Address:	(value- 1234 New Delhi)
Phone:	(value-0116573891)
Director:	Dr Mike

Heart Hospital Frame Instance

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

- **Graphical representation of a frame network** for the class *'hospital'* for the above description is:

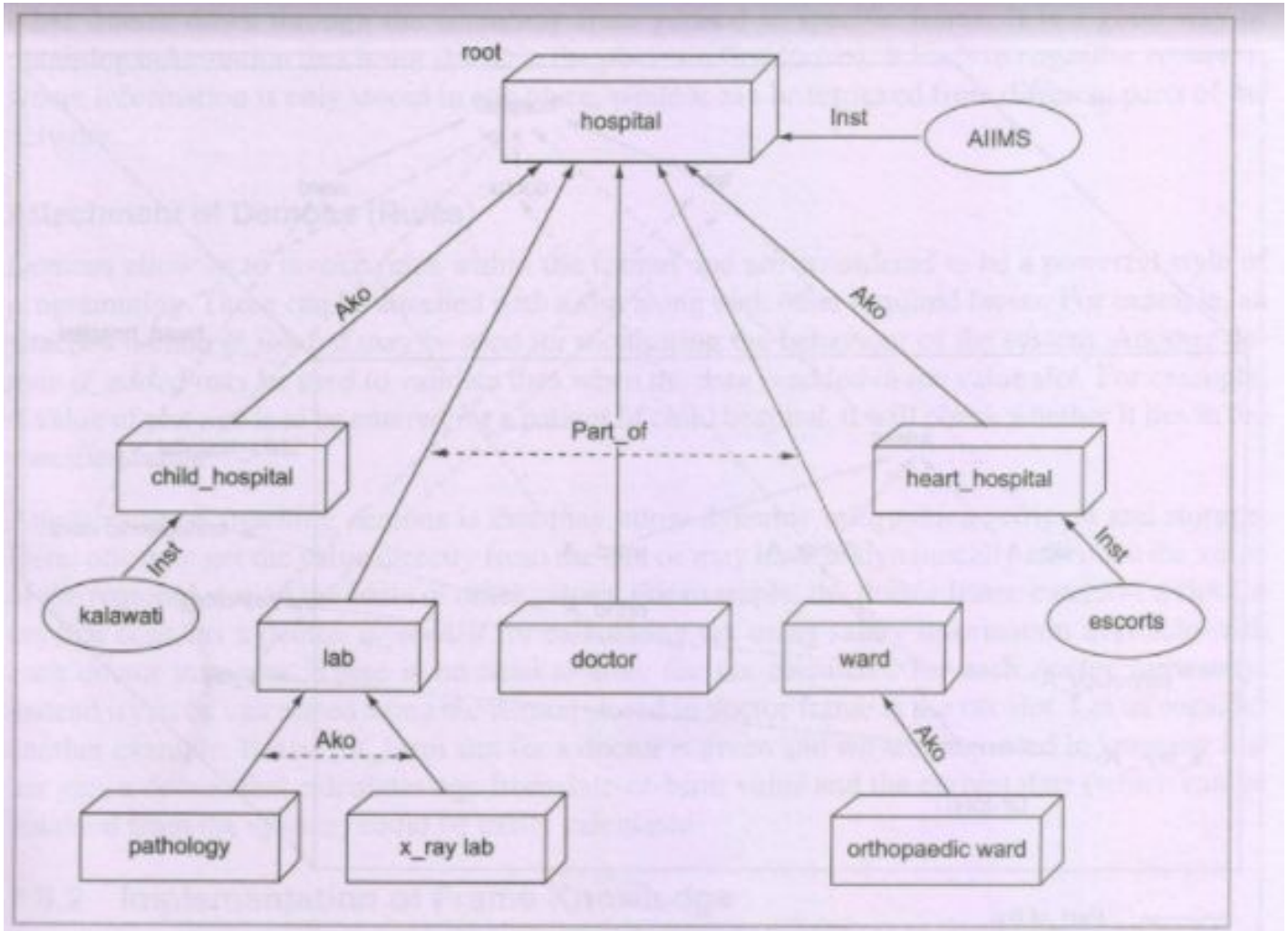


Figure 7.19 A Simple Frame System

- 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.*
  - The second dimension in the hierarchy is of *Ako link.*
  - The third dimension contains the instances of all the classes concerned with *Inst links.*

Note: Each frame can have at most two links with the following combinations:

- A frame can be an instance of some frame and a part of another frame. So, it can have both *Inst* and *Part_of* links.
- A frame can be a-kind-of frame of one frame and a part of another frame. So, it can have *Ako* and *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* and *Ako* in a frame are not possible.

- The three-dimensional view of the *hospital frame structure*:

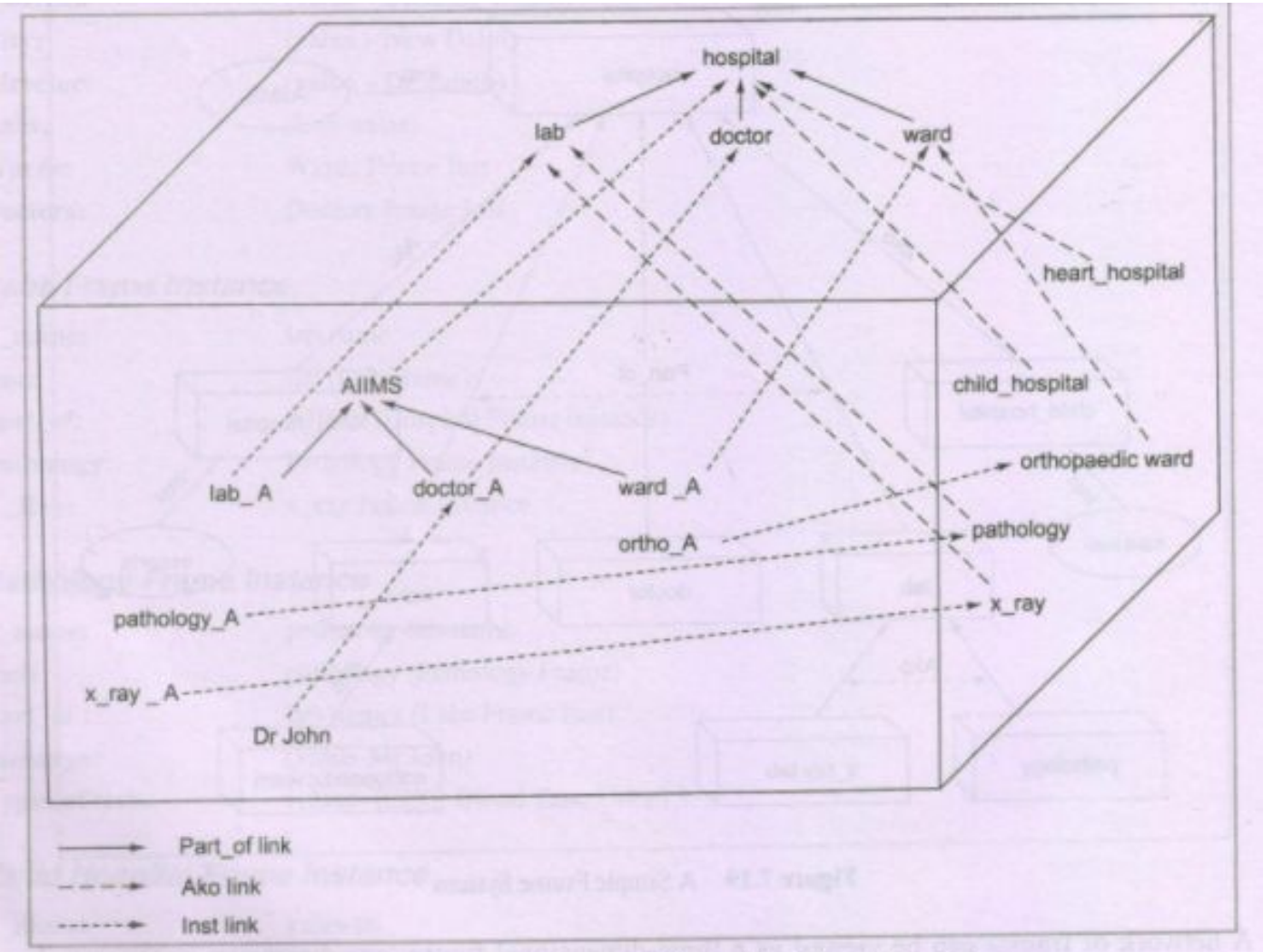


Figure 7.20 Three-dimensional View of the Class 'Hospital'

- An instance of a hospital is a specific hospital, such as AIIMS, which will have instance of all the classes in the network of hospital frame, i.e., instances of the *lab, doctor, ward.*

- Representation of the 3D structure of a network of frames in FOL:

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

where, frame(X) means that X is a frame, $\text{slot}_1 \in \{\text{Inst, Ako}\}$; $\text{slot}_2 = \text{Part\_of}$, 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 network.

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.

Example:

i. *if_needed* (calculating tax using salary information of doctor)
ii. *if_added* (age of patient lies in specified range)

Implementation of Frame Knowledge

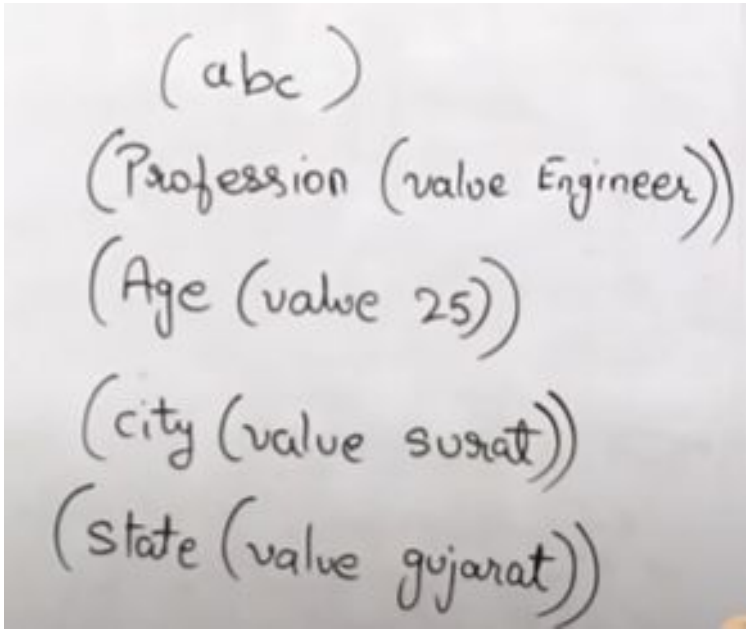
- Frames can be easily implemented using OOP languages.
- The concepts of a class in OOP can be directly used to code frames that has the concept of subclass(Ako) and containment (Part_of_link) of a class within another class.
- Instances of the frames can be declared as objects of the relevant classes.
- Slots can be defined variables and procedural attachment's 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, [slot$_1$(facet(value),….), slot$_2$(facet(value),….),….])

Example:



Representation of Frames in Prolog

Example:

- Representation of part of the network of frame system for hospital in Prolog:

```
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)]).
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,
                urine])] ).
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

- Writing of Inheritance rules helps to obtain relevant information either directly available in the frame or through inheritance.

Example:

- To find the country of 'escort hospital' which is not given in the frame directly – travel starting from escort upward till root to know that country of escort hospital is India.

Inheritance rules for frames might look like:

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

Advantages of Frame Representation:

- The frame knowledge representation makes the programming easier by grouping the related data.
- The frame representation is comparably flexible and used by many applications in AI.
- It is very easy to add slots for new attribute and relations.
- It is easy to include default data and to search for missing values.
- Frame representation is easy to understand and visualize.

Disadvantages of Frame Representation:

- In frame system inference mechanism is not be easily processed.
- Inference mechanism cannot be smoothly proceeded by frame representation.
- Frame representation has a much generalized approach.