

# **UNIT - IV**

# **Artificial Intelligence**

**III / II CSE, R 16 - JNTUK**

**Dr. T. K. Rao**

**VVIT**

# 1. Approaches to Knowledge Representation (KR)

- AI programs use structures (knowledge structures) to represent objects, facts, relationships, and procedures
- Knowledge structures provide expertise and information so that a program can operate in an intelligent way
- KSs are usually composed of traditional and complex structures like semantic network, frames, scripts, conceptual dependency, etc.

- A knowledge base is a special type of function of DB that holds knowledge of the domain
- Knowledge bases share some of the functions such as storing, updating, retrieving info.
- They are much powerful and more structured
- Following are the knowledge representation schemes
  - **Relational knowledge**
  - **Knowledge represented as logic**
  - **Procedural knowledge**

# 1.1 Relational knowledge

- This comprises objects consisting of attributes and associated values
- Simplest way of storing facts is to use a relational method
- A table is defined as a set of data values using rows and columns
- Columns are attributes & rows are corresponding values of attributes

Name	Age (in years)	Gender	Qualification	Salary (in Rs.)
John	38	M	Graduate	20000
Mike	25	M	Undergraduate	15000
Mary	30	F	Doctorate	50000
James	29	M	Graduate	20000

- This representation helps in storing the facts but gives little inference
- It is easy to obtain to answers of the following queries
  - What is the age of John
  - How much does Mary earn
  - What is the qualification of Mike
- But we cannot obtain answers for the questions like “Does a doctorate earn more?”
- So, inferencing new knowledge from structures is not possible

## 1.2 Knowledge represented as logic

- Inferential capacity can be achieved if knowledge is represented in the form of formal logic
- E.g. knowledge about mortality “all humans are mortal” cannot be represented using relational approach
- It can easily be represented in predicate logic
  - $(\forall X)\text{human}(X) \leftarrow \text{mortal}(X)$
- Advantages: a set of rules can be represented, derived more facts, truths and verified the correctness of new statements

# 1.3 Procedural knowledge

- This is encoded in the form of procedures which carry out specific tasks
- E.g. an interpreter interprets the program on the basis of available knowledge regarding the syntax and semantics of the language
- Advantages: domain specific knowledge can be easily represented and side affects can be easily modelled
- Disadvantage: problem in completeness and consistency

## 2. KR using semantic network

- Basic idea is, the meaning of the concept is derived from its relationship and other concepts
- The information is stored by interconnecting nodes with labelled arcs
- E.g. 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.



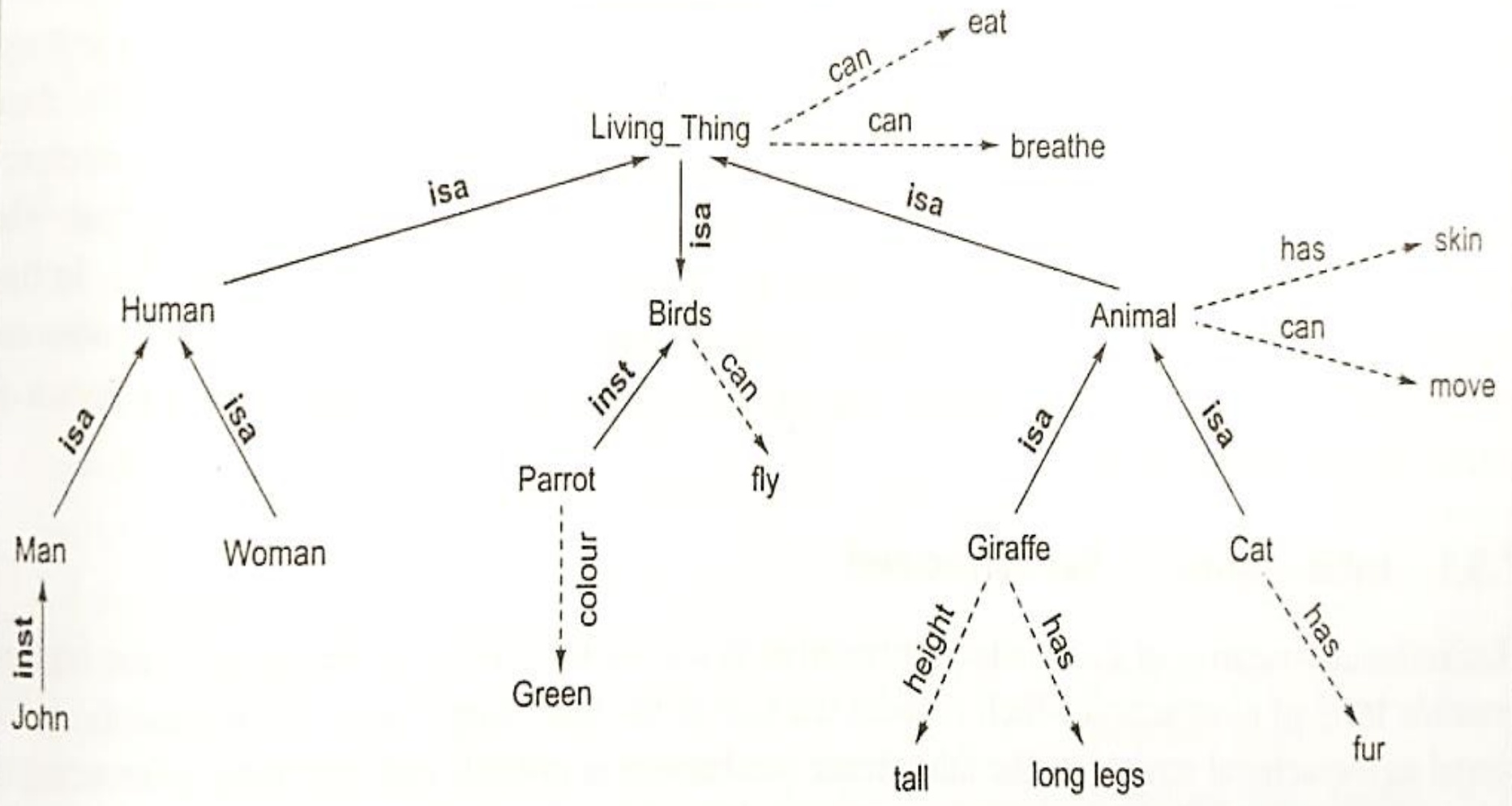
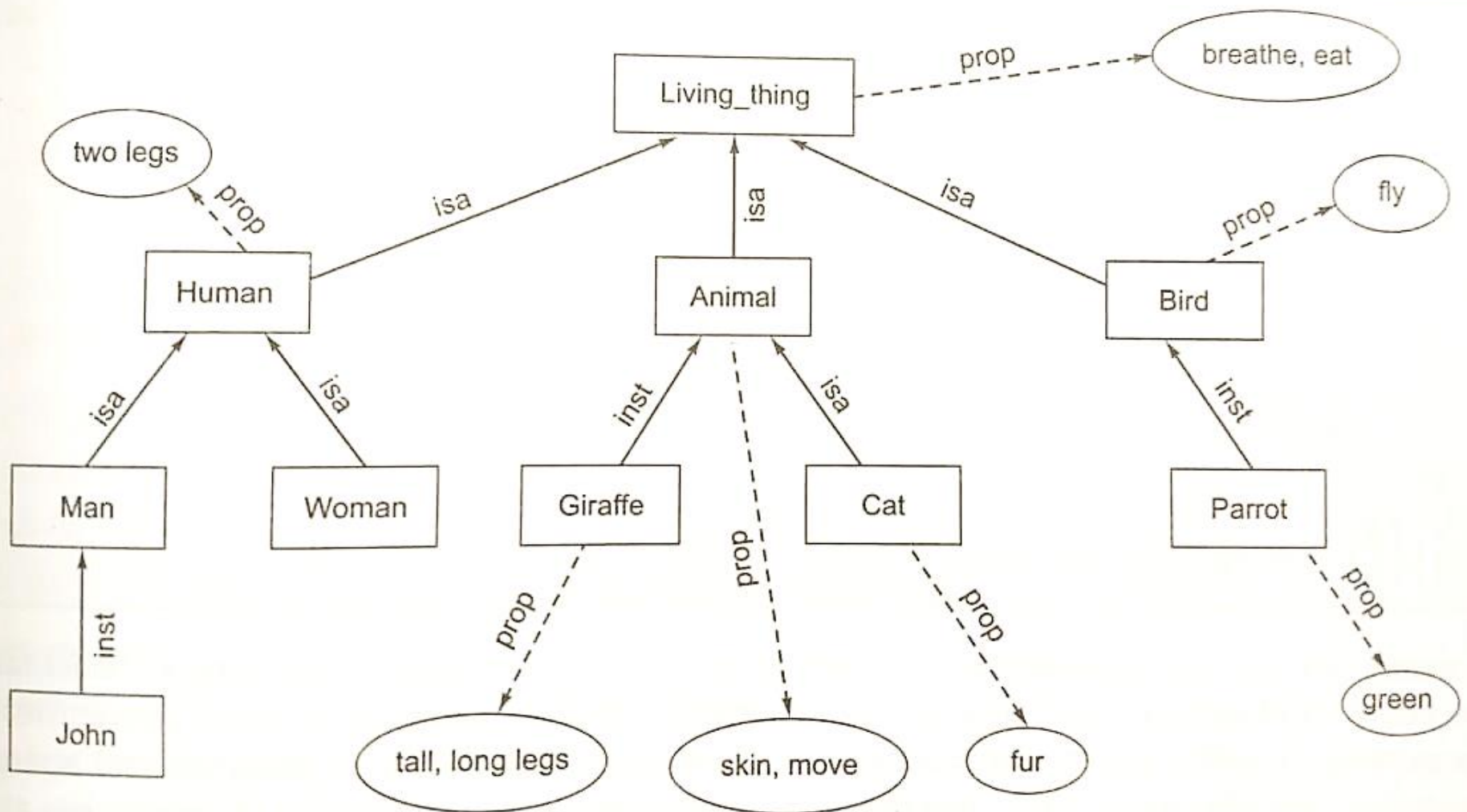


Figure 7.1 Knowledge Representation using Semantic Net

- isa – relation connects two classes where one concept is a kind or subclass
- inst- relation relates specific members of a class, such as John is an instance of Man
- Other relations such as {can, has, colour, height} are known as property relations
- These are represented in dotted lines pointing from the concept to property
- In this structure, property inheritance is easily achieved
- E.g. the query “does a parrot breathe?” can be easily answered as ‘yes’



**Figure 7.2** Concepts Connected with *prop* Links

- ***isa*** and ***inst*** links have well-defined meaning
- Semantic net interpreter cannot understand all the attribute links unless their semantics are encoded into it
- The interpretation of a prop link can be the property of the class
- The information encoded can be represented as a digraph as in fig. 7.2
- The square nodes represent concepts or objects connected with ***isa*** or ***inst*** links
- Oval nodes represent property attributes attached to the square nodes

## 2.1 Inheritance in semantic net

- Hierarchical structure of KR allows knowledge to be stored at the highest possible level of abstraction
- This reduces the size of the knowledge base
- Semantic net is stored as hierarchical structure hence, inheritance mechanism is in-built
- It helps in maintaining consistency of knowledge base by adding new concepts and members to existing ones
- The algorithm to retrieve a property of an object is as follows

# Algorithm: Property Inheritance Algorithm

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

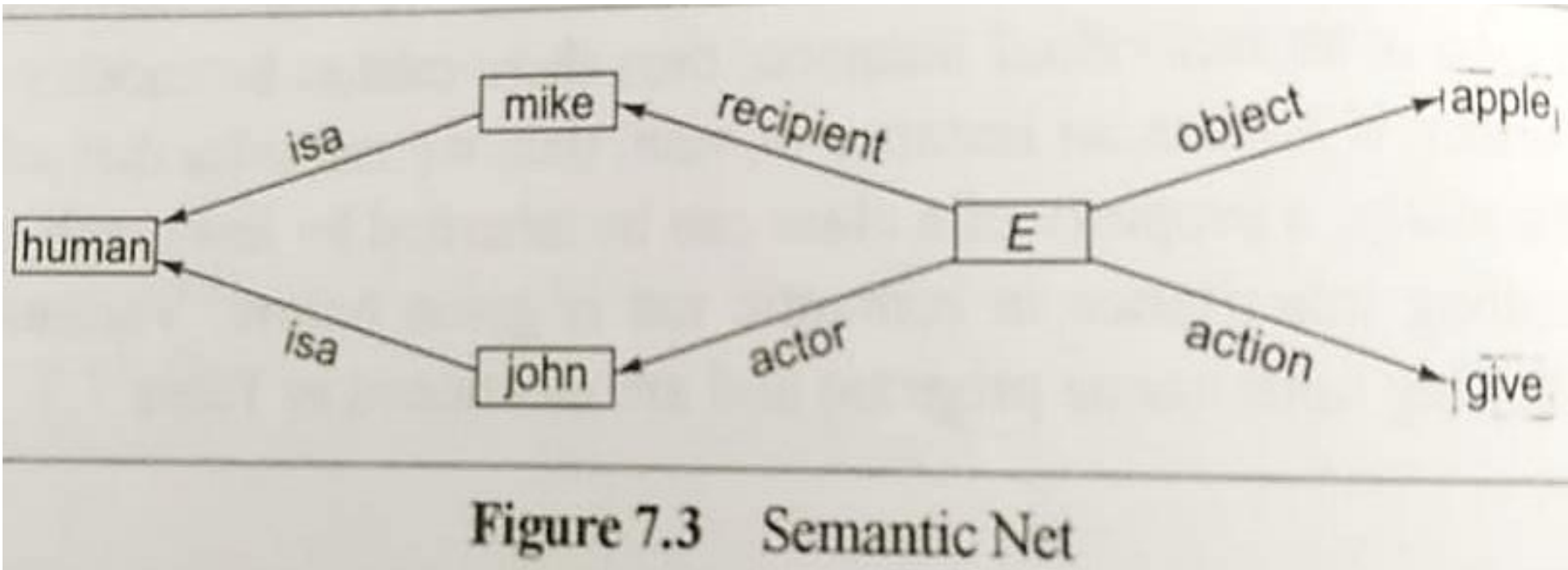
**Output:** returns 'Yes', if object has the desired property else returns 'No'

**Procedure:**

- Find an object in the semantic net
- Found = false
- While [(object != root) or found] Do
  - {
  - If there is an attribute attached with object then found = true
  - Else {Object = isa(object, class) or object = inst(object, class)}
  - }
- If found = true then report 'yes' else report 'no'

### 3. Extended Semantic N/Ws for KR

- Simple semantic n/w is represented as a digraph whose nodes represent concepts/objects & arcs represents relations between concepts/objects
- An alternative way of representing semantic net in detail with more semantic links is Extended Semantic N/Ws
- The English sentences “John gives an apple to Mike and John and Mike are Human” may be represented in semantic n/w as follows



- Here E represents an Event which is an act of giving, whose actor is John, the object is an apple, and recipient is Mike
- Semantic net can hold semantic information about situation such as actor of an event giving is john and object is apple
- In the sentence, John gives an apple to Mike



- The relations of 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)
- Predicate relations of labels on the arcs of semantic n/ws always have two arguments
- Hence the entire semantic net can be coded using binary representation (two argument reptn.)

- Such representation is advantageous when additional information is added to the system
- E.g. John gave an apple to Mike in the kitchen, it is easy to add *location(E, kitchen)* to the set of above facts
- In **First Order Predicate Logic**, predicate relation can have 'n' arguments, where  $n \geq 1$
- E.g. John gives an apple to Mike is represented in predicate logic by *give(John, Mike, apple)*
- Here John, Mike and apple are arguments, ***give*** represents a predicate relation

- Predicate logic representation has greater advantages compared to semantic net
- E.g. John gives an apple to everyone he likes
  - The sentence is expressed in predicate logic by the clause as
$$\textit{give}(\textit{John}, X, \textit{apple}) \leftarrow \textit{likes}(\textit{John}, X)$$
  - here the symbol X is a variable representing any individual
  - here left side of ‘<-’ contains conclusions, while right side of ‘<-’ contains conditions

- Disadvantage: it is not convenient to add new information in an n-ary representation
- E.g. if we have 3-ary relation, *give(John, Mike, apple)* representing 'John gives an apple to Mike'
  - To capture new information about kitchen in the sentence 'John gave an apple to Mike in kitchen',  
*give(John, Mike, apple, kitchen)*

- 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
- E.g. if John gives something he likes to a person, then he also likes that person
  - This is represented as

*likes(John, X) <- give(John, X, Y), likes(John, Y)*

- In conventional semantic n/w, we cannot express clausal form of logic
- It can be surpassed by Extended Semantic N/w (ESNet), that combines the advantages of both Logic and Semantic
- ESNet can be interpreted as
  - a variant syntax for the clausal form of logic,
  - It has the same expressive as that of predicate logic with well-defined semantics, inference rules
  - A procedural interpretation
  - Also incorporates the advantages of using binary relation as a semantic n/w rather than n-ary relations of logic



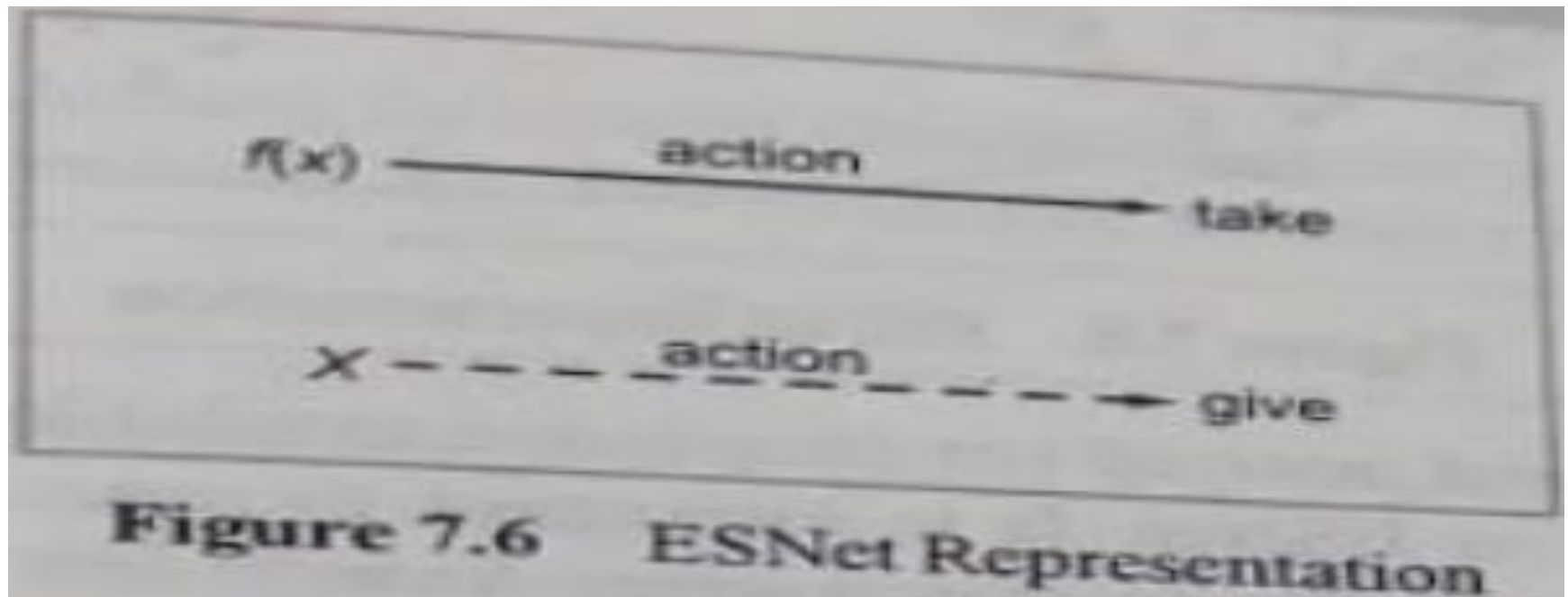
**Figure 7.4** ESNet Representation



**Figure 7.5** ESNet Representation

## 3.1 Inference rules

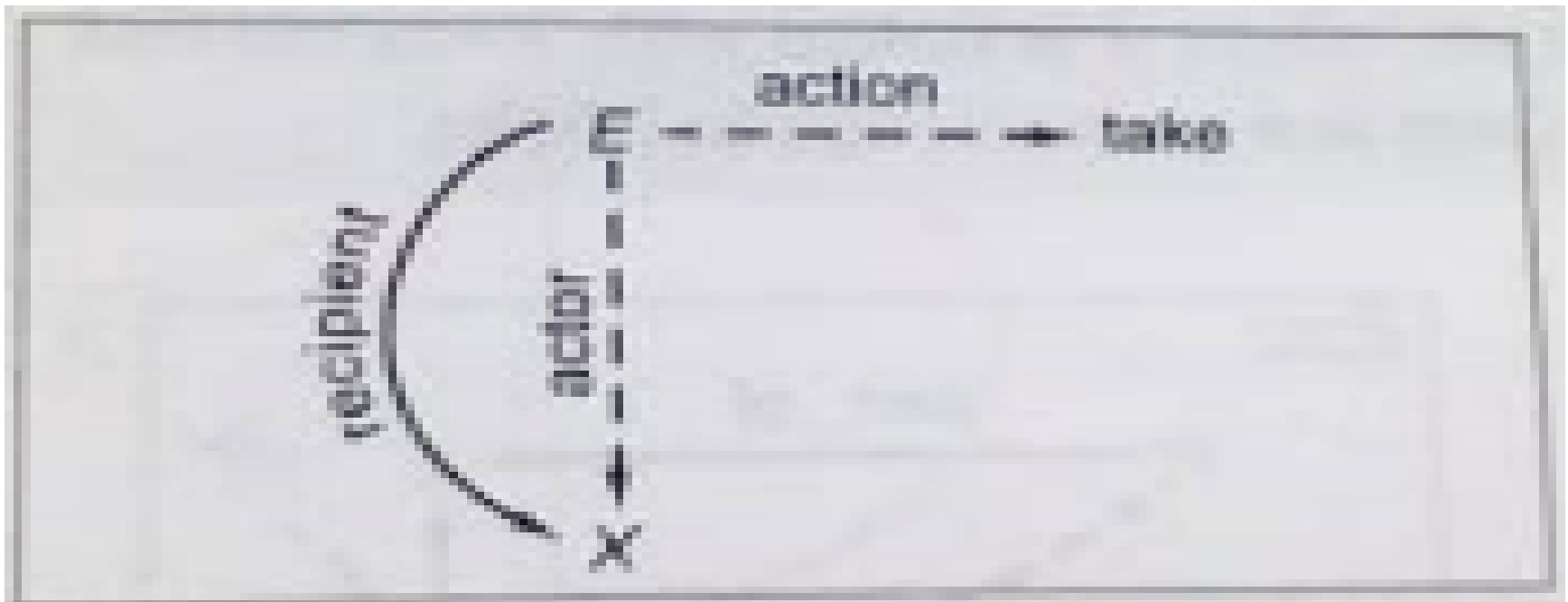
- Some of the rules are
  - The representation of the inference for every action of giving, there is an action of taking in clausal logic ***action(f(x), take) <- action(x, give)***





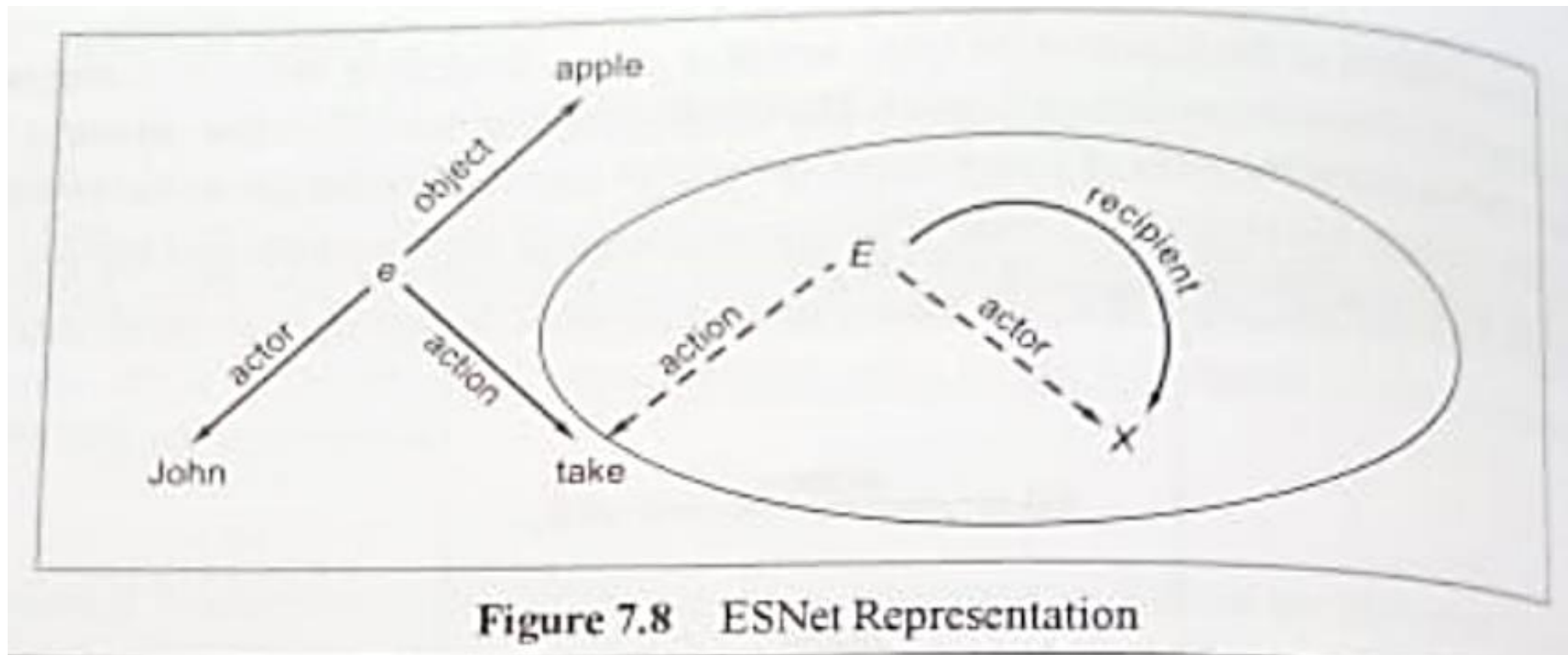
- The inference rule that an actor performs a taking action is also the recipient of this action and can be easily represented in clausal logic

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



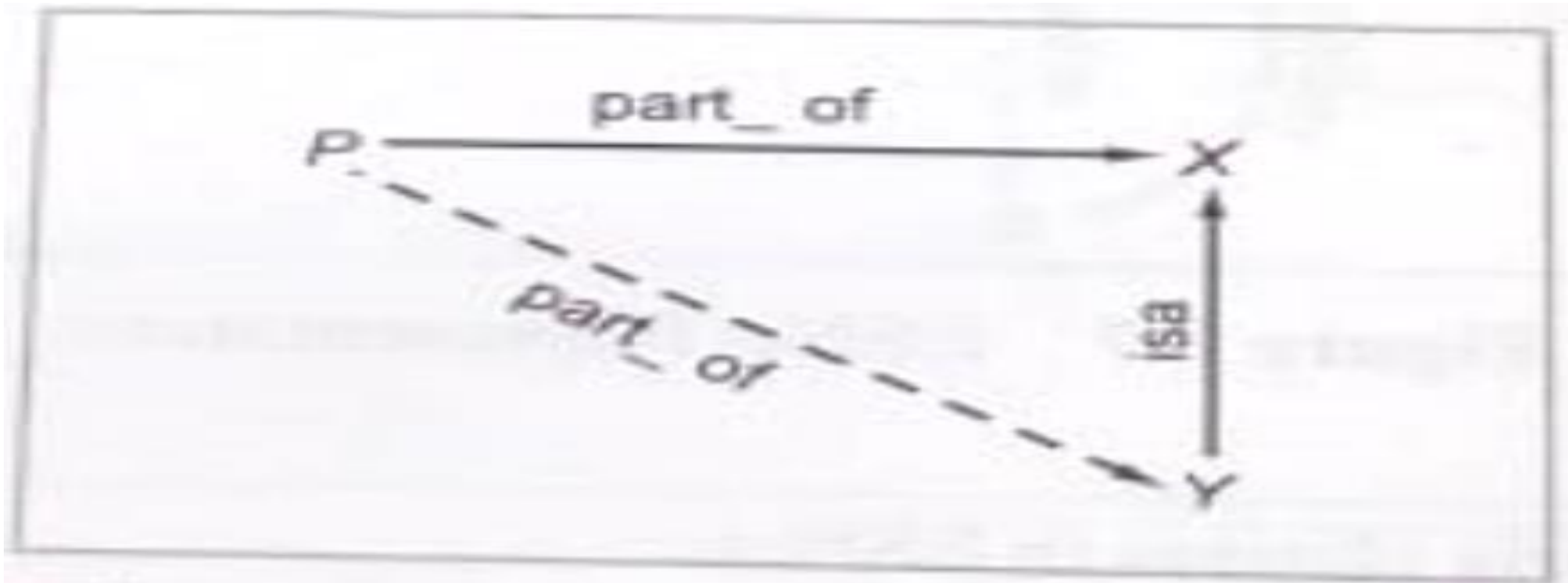
**Figure 7.7 ESNet Representation**

- E.g. 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)



**Figure 7.8** ESNet Representation

- the hierarchy links *isa*, *inst* and *part\_of* (or *prop*) are available in semantic n/ws also as special cases 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



**Figure 7.9** Contradiction in ESNet

## 3.2 Deduction in ESNeTs

- In logic there are two types of mechanisms
  - Forward reasoning inference mechanism  
(also called bottom-up approach)
  - Backward reasoning inference mechanism  
(also called top-down approach)

# Forward reasoning inference mechanism

- Given an ESNet, apply the following reduction using Modus Ponens rule of logic (i.e., given ***(A $\leftarrow$ B) and B, then conclude A***)
- E.g. ***isa(X, human)  $\leftarrow$  isa(X, man)***  
***isa (John, man)***
- Using Modus Ponens rule of logic, ***isa(John, human)*** become true

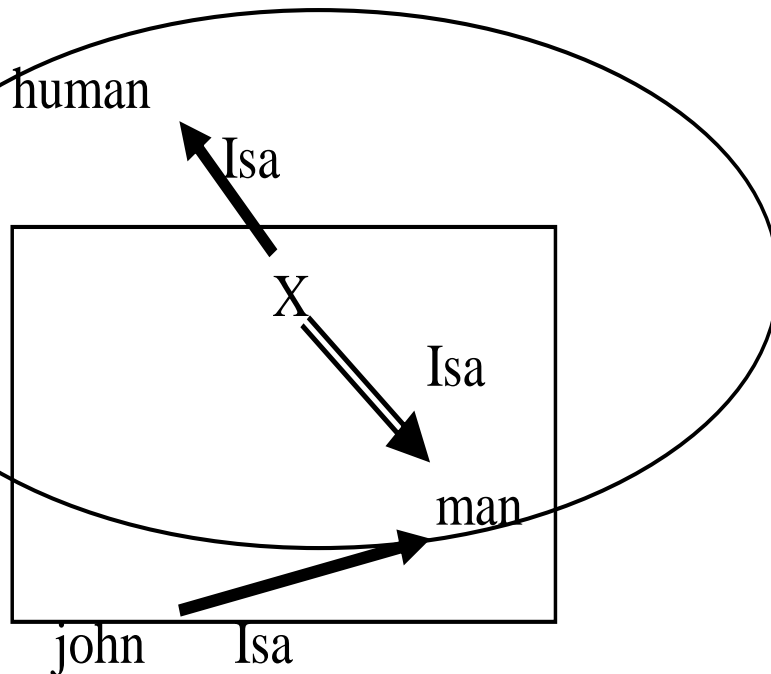
# Forward reasoning inference

## Given set of clauses

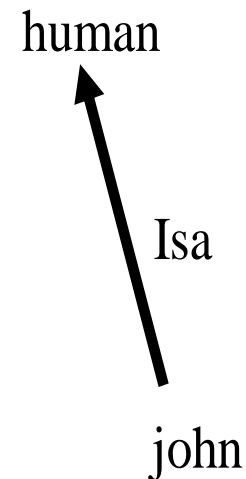
$\text{Isa}(X, \text{human}) \leftarrow \text{Isa}(X, \text{man})$   
 $\text{Isa}(\text{john}, \text{man}).$

## Inferencing

$\text{Isa}(\text{john}, \text{human})$



Here X is bound to john



# Backward reasoning inference mechanism

- In this, a conclusion or goal can be proved from a given ESNet by adding the denial of the conclusion to the n/w and show that the resulting set of clauses in the n/w gives contradiction
- We need to prove *isa(John, human)* using the same network

# Backward reasoning inference

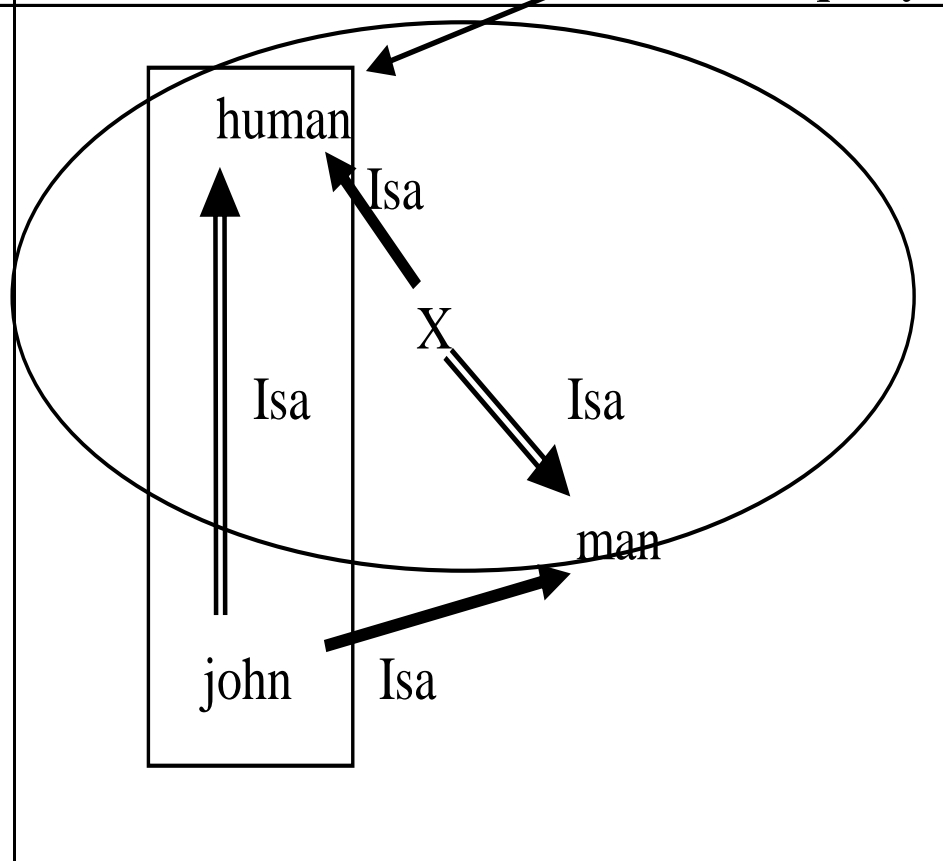
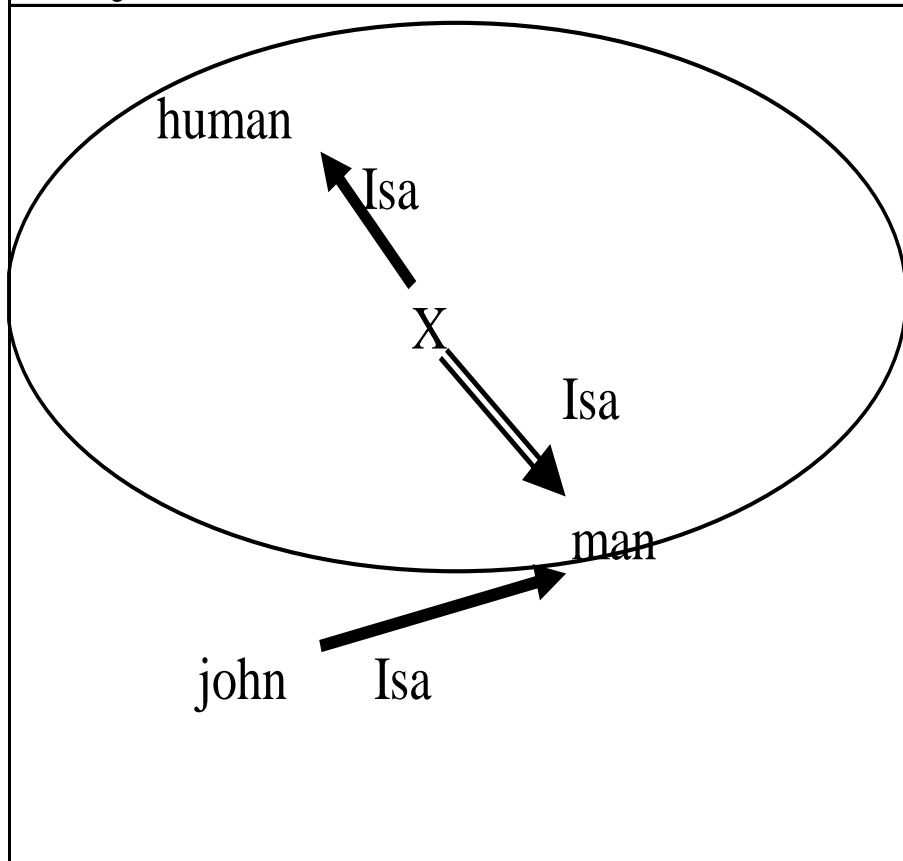
## Given set of clauses

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

## Prove conclusion

Query:  $\text{Isa}(\text{john}, \text{human})$

denial of query





## 4. KR using Frames

- Frames are an extension to semantic nets
- Each node of a semantic net is represented by a Frame, which can be defined as a DS
- It consists of collection of attributes or slots and associated values that describe real-world entity
- Frames are slightly similar to the O-O paradigm
- Frames consists of attributes or slots, which are described with attribute-value pairs  
<slot\_name, value>

- Slots are complex structures in general, that have facets (or fillers) describing their properties
- Value of a slot is text, integer, constant or it may be another frame
- So slots may contain value, refer to other frames or methods
- 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).

# Structure of a frame

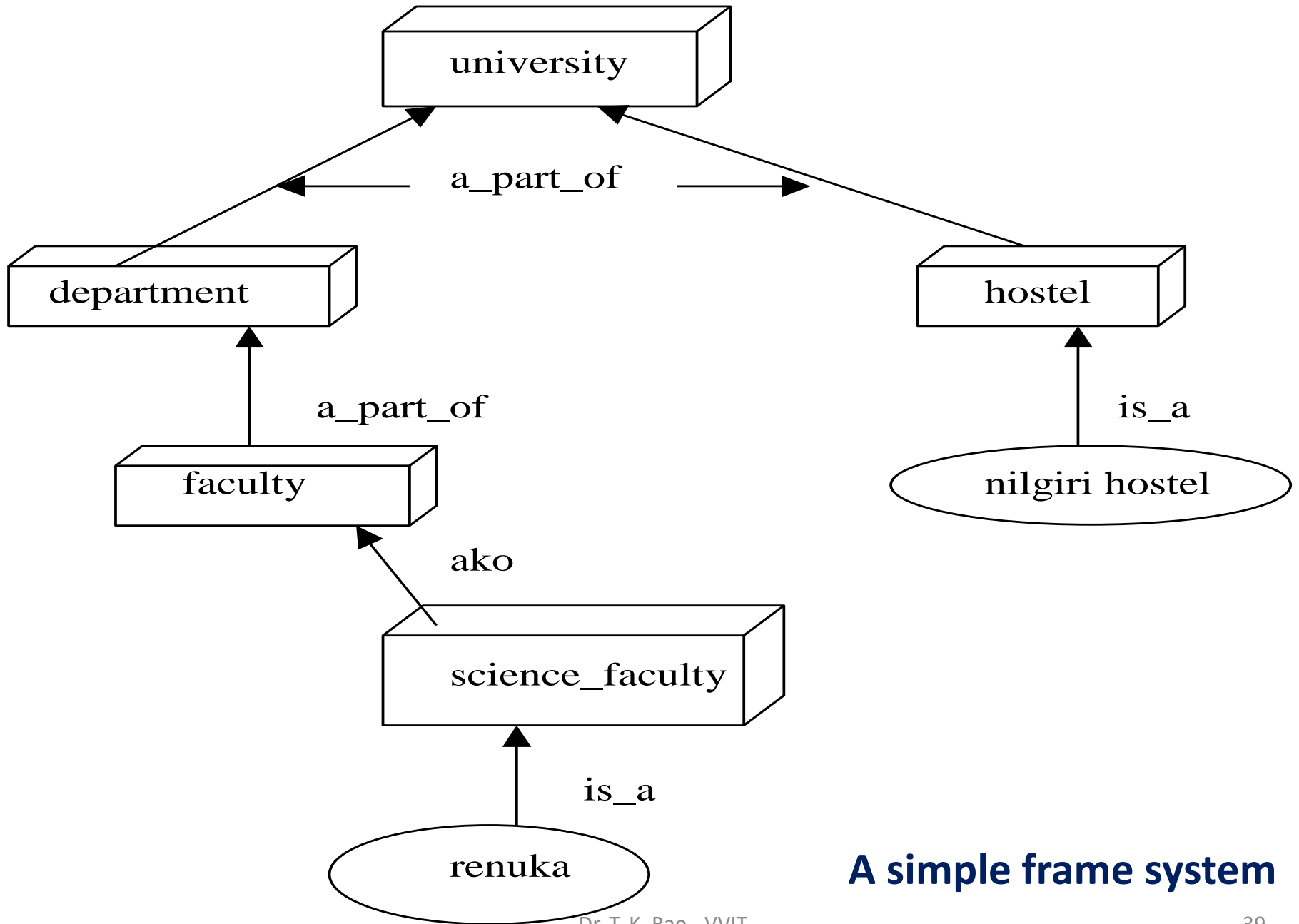
Frame name
Slot-filler
Default values
Constraints on values within the slots of a frame
Pointers (links) to other frames
Ako (a-kind-of or subclass)
Inst (instance)
Instantiation procedure
Inheritance procedure
Default inference procedure
Triggers

Slot	Value	Type
ALEX	—	(This Frame)
NAME	Alex	(key value)
ISA	Boy	(parent frame)
SEX	Male	(inheritance value)
AGE	IF-NEEDED: Subtract(current,BIRTHDATE);	(procedural attachment)
HOME	100 Main St.	(instance value)
BIRTHDATE	8/4/2000	(instance value)
FAVORITE_FOOD	Spaghetti	(instance value)
CLIMBS	Trees	(instance value)
BODY_TYPE	Wiry	(instance value)
NUM_LEGS	1	(exception)

Slot	Value	Type
BOY	—	(This Frame)
ISA	Person	(parent frame)
SEX	Male	(instance value)
AGE	Under 12 yrs.	(procedural attachment - sets constraint)
HOME	A Place	(frame)
NUM_LEGS	Default = 2	(default, inherited from Person frame)

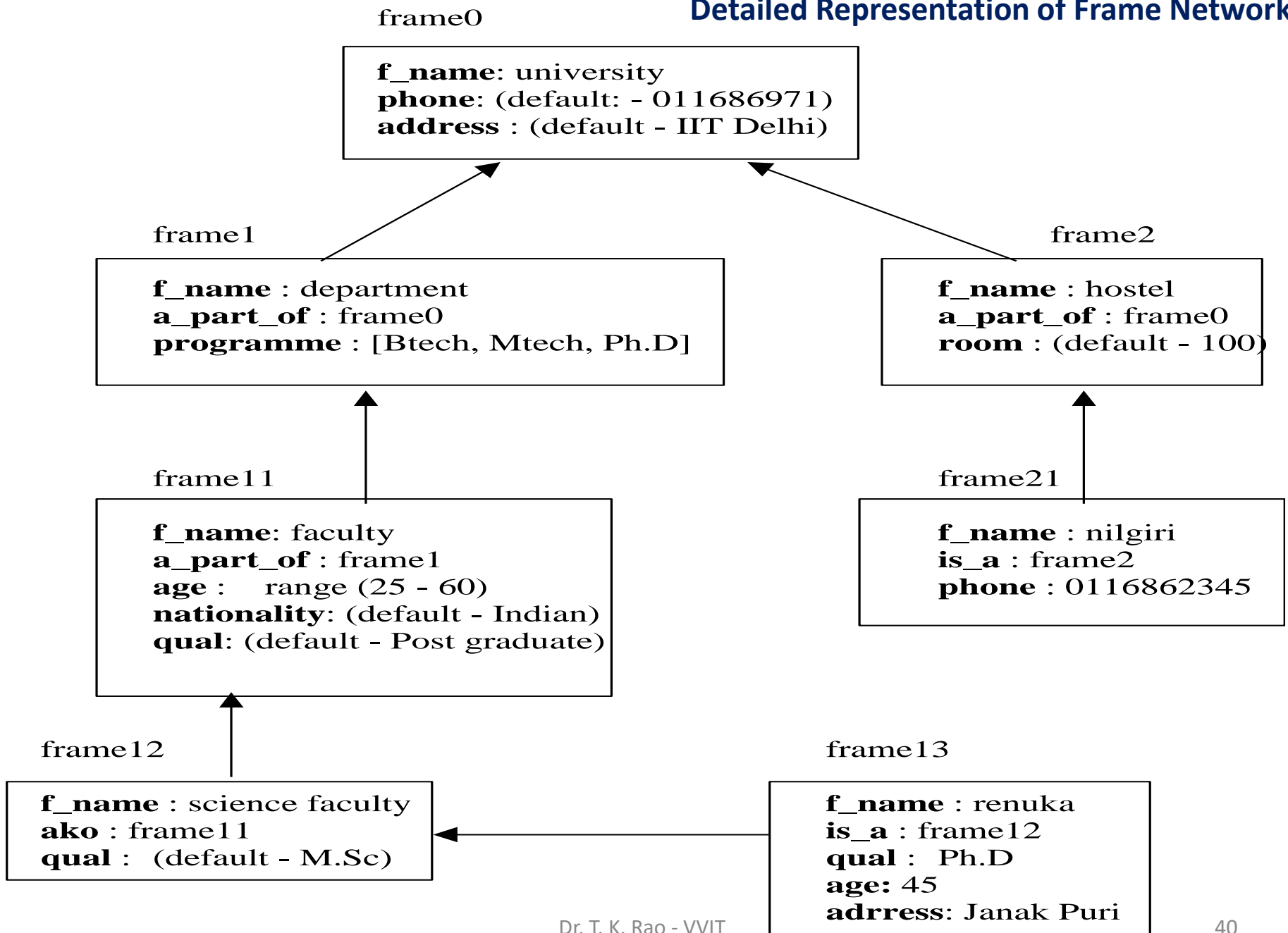
Slot	Value	Type
MONKEY	—	(This Frame)
ISA	Primate	(parent frame)
SEX	OneOf(Male,Female)	(procedural attachment)
AGE	an integer	(procedural attachment - sets constraint)
HABITAT	Default = Jungle	(default)
FAVORITE_FOOD	Default = Bananas	(default)
CLIMBS	Trees	—
BODY_TYPE	Default = Wiry	(default)
NUM_LEGS	Default = 2	(default)

- Frame contains information on how to use frame, what to expect next, & what if expectations are not met
- A frame's terminals are already filled with default values, which is based on how the human mind works
- Strength of frame based KR is that, unlike semantic networks, they allow exceptions in particular instances
- This gives frames an amount of flexibility that allow representations of real world phenomena to be reflected more accurately



## A simple frame system

## Detailed Representation of Frame Network





- A frame can be an instance of some frame and a part of another frame, hence it can have both ***Inst*** and ***Part\_of*** links
- A frame can be a\_kind\_of frame of one frame and apart of another frame, hence, it can have ***AKO*** and ***Part\_of*** links
- It is not possible to have a frame which is both an instance and a\_kind\_of some class at the same time, hence the links ***Inst*** and ***AKO*** in a frame are not possible

## 4.1 Inheritance in frames

- Inheritance is a mechanism which is utilized for passing knowledge from one frame another
- It is a good way of obtaining information that is not stored in the place we first looked
- It leads to cognitive economy, where info. is stored at one place, while it can be retrieved from different parts of the n/w

# Attachment of Demons

- 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
- E.g. A demon ***if\_added*** may be used to validate data when added in the value slot
  - If value slot is to be entered for a patient of child hospital, it will check whether it is in the specified range or not

# 5. Conceptual Dependency Theory

- Conceptual Dependency (CD) theory is used for natural language understanding
- It focuses on the concept and meaning rather than on syntax and structure
- CD uses the basic hypothesis that the ACTION is the basis of any proposition
- Propositions that describe events are made up of conceptualizations that include an actor, an action & few cases that depends on the action

- Similarities and overlapping in meanings are handled by the concept PRIMITIVE ACTION
- The primitives are the elements that can be used in varied combinations to express the underlying meaning of the given word
- CD uses 11 primitive acts, proposed by Schank
- The structural similarities in the sentence represented in actor-action-object framework

# 5.1 Conceptual Primitive Actions

- Actions in natural language are generally specified by the verbs
- Most of the verbs are categorized under the following eleven primitive actions
- Sometimes one verb may fall into two categories
- Then the most relevant in the context might be chosen for interpreting the sentences
- The PRIMITIVE ACTIONS are as follows:

1. ATRANS
2. PTRANS
3. PROPEL
4. MOVE
5. GRASP
6. INGEST
7. MTRANS
8. MBUILD
9. EXPEL
10. SPEAK
11. ATTEND

- **ATRANS:** Transfer of an abstract relationship such as possession, control, or ownership, e.g. give, take, buy, etc.
- It requires an actor, object and recipient
- **PTRANS:** Transfer of the physical location of an object or actor that requires an actor, object, and direction, e.g. go, walk, fly, etc.
- **PROPEL:** Application of physical force to an object, i.e. push, pull, etc.
- **MOVE:** Movement of a body part by its owner, e.g. kick, throw
- **GRASP:** Grasping of an object by an actor, e.g. catch, clutch, etc.
- **INGEST:** Ingesting of an object by an animal, e.g. eat, drink, smoke, etc.
- **MTRANS:** Transfer of mental information between animals or within an animal itself, e.g. read, tell, speak, sing, etc.

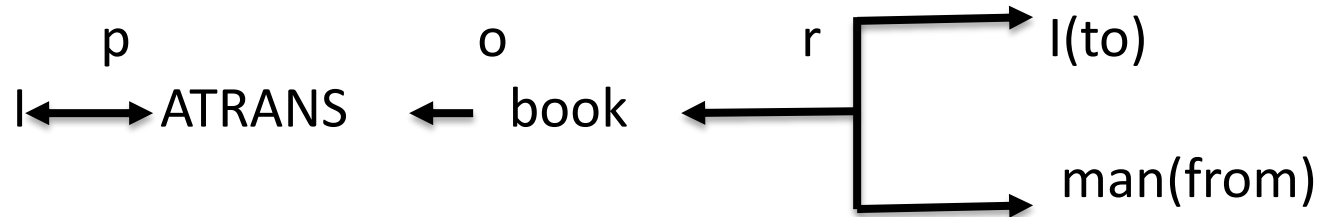


- **MBUILD:** The construction of new information from old information within actor, i.e. to create or combine thoughts internally, e.g. decide, describe, imagine, consider, answer, etc.
- **EXPEL:** Expulsion of something from the body of an animal, e.g. weep(expel tears), cry(expel tears), swear(expel moisture), etc.
- **SPEAK:** Producing sounds, e.g. say, tell, sing, etc.
- **ATTEND:** Focusing of a sense organ toward a stimulus, e.g. listen, watch, see, etc.

## 5.2. Conceptual Category

- CD provides a specific set of building blocks from which representations can be made
- There are four primitive conceptual categories from which dependency structures can be built
  - **ACT:** Actions {one of the CD primitives representing verb}
  - **PP:** Objects {pictures producers or noun/pronoun}
  - **AA:** Modifiers of actions {action aiders or adverb}
  - **PA:** Modifiers of PPs {picture aiders or adjective}

- The relationships between concepts are called dependencies
- The main conceptualization of a clause is a two-way dependency b/w a PP & an action
- Actions are broken down into sequence of primitive ACTs
- E.g. the CD representation for sentences such as “I took a book from the man”, “The man gave me a book”, “The book was given by a man to me” having the same meaning



- Here
  - o – for the object case relation
  - r – for the recipient case relation
  - d – destination
- Few other conceptual tenses are
  - P- past
  - F-future
  - T-transaction
  - ts - start transaction
  - Tf – finish transaction
  - K – continuing
  - C - conditional

## 5.3. Rules for Conceptualization Blocks in CD

- The dependencies among conceptualisation correspond to semantic relations
- Following is the list of the most important structures or rules:
  - Rule1: relation b/w an actor and event caused by him is called a two-way dependency, as neither actor not event can be considered primary

## Rule 1: PP $\Leftrightarrow$ ACT

Examples	CD Representation
John ran	$\overset{p}{\text{John}} \Leftrightarrow \text{PTRANS}$
Mary cried	$\overset{p}{\text{Mary}} \Leftrightarrow \text{EXPEL}$

Rule 2: Rule representing relation b/w an ACT and the PP (object) of the ACT is shown by the direction of an arrow toward the ACT, since the context of the specific ACT determines the meaning of the object relation

## Rule 2: ACT $\leftarrow$ PP

Examples	CD Representation
John gave a book to Mary	$\text{John} \Leftrightarrow \text{PTRANS} \leftarrow \text{book}$
Mary kicked the ball	$\text{Mary} \Leftrightarrow \text{MOVE} \leftarrow \text{ball}$

- Rule 3: This rule shows relation both ways b/w two PPs, one belong to the set defined by the other PP

Rule 3: PP $\Leftrightarrow$ PP	
Examples	CD Representation
John is a doctor	John $\Leftrightarrow$ doctor
Mary is a teacher	Mary $\Leftrightarrow$ teacher

- Rule 4: It shows a relation b/w two PPs, one of the PP provides a particular kind of information about the other PP

Rule 4: PP $\leftarrow$ PP	
Examples	CD Representation
John's car	<div>poss-by</div> <div>Car <math>\leftarrow</math> John</div>
Mary is in Delhi	<div>loc</div> <div>Delhi <math>\leftarrow</math> Mary</div>

- Rule 5: It shows a relation b/w a PP and a PA that is asserted to describe it

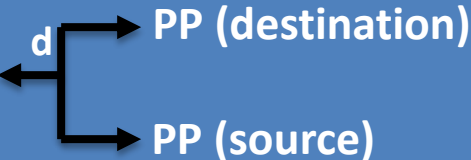

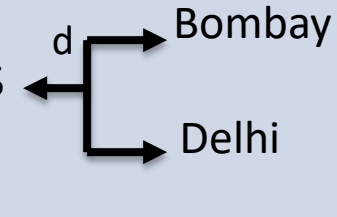
Rule 5: PP $\Leftrightarrow$ PA	
Examples	CD Representation
John is fat	John $\Leftrightarrow$ weight ( $>50$ )
Mary is poor	Mary $\Leftrightarrow$ income ( $<200$ )

- Rule 6: It shows a relation b/w a PP and an attribute that already has been prediction of it

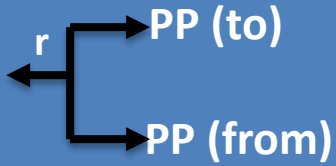

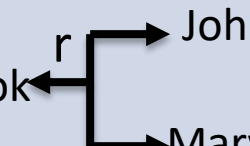
Rule 6: PP $\Leftrightarrow$ PA	
Examples	CD Representation
Smart John	John $\leftarrow$ smart
Charming Mary	Mary $\leftarrow$ charming



- Rule 7: it shows a relation b/w an ACT and its physical source and destination locations of ACT

<p>Rule 7: ACT </p>	
Examples	CD Representation
John went to school from home	<p>John <math>\leftrightarrow</math> PTRANS </p>
Mary travelled from Delhi to Bombay by train	<p>Mary <math>\leftrightarrow</math> PTRANS </p> <p style="margin-left: 150px;">v ↑ train</p>

- Rule 8: it shows a relation b/w an ACT and its source and the recipient of ACT

<p><b>Rule 7: ACT</b></p> 	
Examples	CD Representation
John gave a book to Mary	<p> <math>p</math>                      <math>o</math>                      <math>r</math> </p> <p>       John <math>\Leftrightarrow</math> ATRANS <math>\leftarrow</math> book <math>\leftarrow</math>  </p>
Mary told story to John	<p> <math>p</math>                      <math>o</math>                      <math>r</math> </p> <p>       Mary <math>\Leftrightarrow</math> MTRANS <math>\leftarrow</math> book <math>\leftarrow</math>  </p>

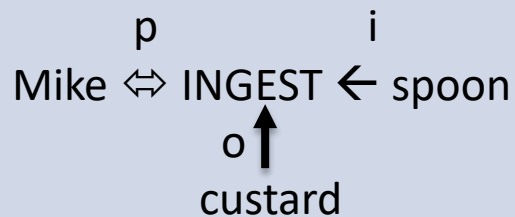
- Rule 9: It shows a relation b/w an ACT and the instrument using which it is performed

### Rule 9: ACT $\xleftarrow{i}$ detailed conceptualization for instrumental

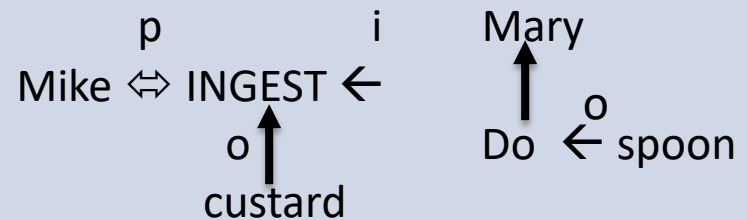
Ex: Mike ate custard with spoon

#### CD representation

##### Simple form



##### Detailed form



## 6. Case Grammars

- Charles J. Fillmore proposed the Case Grammar theory in 1968
- Introduced 6 cases (thematic cases or roles):
  - Agentive (subject)
  - Objective (object)
  - Instrumental (instrument)
  - Dative (covers experiencer)
  - Factive (covers result of an action)
  - Locative (location of action)

- Agent (instigator of action):
  - Noun phrase of the sentence, the instigator of the action
  - E.g. John was given a book, here John is not agent, it may be dative of beneficiary
- Object (identity on which action is performed):
  - The noun being acted upon by action
  - E.g. John broke the door, here door is object
- Dative (animate entity affected by the action):
  - Refers to object of an action that is animate
  - E.g. John killed Mike, here Mike is dative case

- Experienter (animate subject in an active sentence with no agent):
  - For the sentence which have intransitive verbs, the subject become experienter
  - E.g. John cried, here John is experienter
- Beneficiary (animate who was benefited by action):
  - It is an animate person for whom an action is performed
  - E.g. I gave an apple to mike, here Mike is beneficiary

- Locative (place of action):
  - It can be enhanced to Source\_location, Destination\_location, etc.
  - E.g. John went to school, location case is school
- Instrument (entity used for performing action):
  - If some instrument is used to perform an action, then it fills the instrument case
  - E.g. John ate an apple with fork, here fork is instrument
- Co-Agent:
  - If two people perform some action together, the second person fills the Co-agent case
  - E.g. John and Mike lifted the box, here Mike is co-agent

- Time:
  - This case basically contains the time when the action was performed
    - E.g. John went to market yesterday morning, here yesterday and morning are time case
  - Tense (tense of event):
    - Present, past, future of the sentence can be stored in order to generate surface structure sentence
    - E.g. I'm going to school, here present continuous is the tense case
    - Designing grammar describe the relation b/w verbs and their arguments is the main focus in generating case frame



- The mandatory fields are mentioned in Lexicon, whereas optional entries are represented by three dots

Verb lexicon for case grammar
Eat [Agent, Object, (prop - eatable), ...]
Give [Agent, Object, Beneficiary, ...]
Run [Agent, ...]
Want [Agent, Object, ..]
Kill [Agent, Dative, ...]

- The verb eat requires mandatory thematic cases as Agent, Object that is eatable
- It may have optional cases such as instrument, location time, etc.

- Cases are associated with questions about the action and Case Frame is as follows
- E.g. John gave an apple to Mike in the kitchen

Case	←	Question	Answer
Action	←	What was the event	Gave
Agent	←	Who did the action	John
Objective	←	What was involved in the event	Apple
Beneficiary	←	Whom was it done/ who was beneficiary	Mike
Time	←	When was the event done	Past
Location	←	Where was the event done	Kitchen

- Examples of syntactically same but semantically different statements
  - Mike saw the girl in the garden with a **telescope**
  - Mike saw the girl in the garden with a **cat**
  - Mike saw the girl in the garden with a **fountain**
- There are two levels of every sentence structure:
  - Surface structure: a spoken/ written sentence
  - Deep structure: gives underlying meaning of the sentence

# 7. Semantic Web (SW)

- It is an extension of web that provides a common framework, which allows data to be shared and reused across different applications
- SW meaning or semantics to web content to make it easier to be used
- SW have data & documents so that machines can process, transform, assemble, and even act on data in useful ways
- SW comprises XML, XML Schema, RDF, RDF Schema, and Web Ontology Language (OWL)

- **EXtensible Markup Language (XML):**
  - A general purpose specification for creating custom markup languages, like HTML
  - XML tags are not predefined and user may define his own tags
  - It is self-descriptive, Its primary purpose is to facilitate the sharing of structured data across different systems
  - XML was designed to carry data across different information systems, not to display data as in HTML

- XML representation

<circular>

<to> faculty </to>

<from> HoD </from>

<heading> Faculty Meeting </heading>

<body> There is a meeting of the faculty in  
committee room today at 2 p.m </body>

</circular>

- **XML Schema:**

- It is a language for describing the structure of an XML document, typically expressed in terms of constraints on the structure and content of docs.
- They provide means for defining the structure, content and semantics of XML docs. in more detail
- Purpose of XML schema is to define the valid building blocks of an XML document and that consists the following:
  - Elements and attributes that can appear in a document
  - Whether an element is empty or can include text
  - Data types for elements and attributes with default and fixed values for elements and attributes

- **Typical XML Schema:**

```
<xs:schema xmlns:xs=
```

```
http://www.w3.org/2001/XMLSchema>
```

```
</xs:schema>
```

```
<xs:element name = "title" type= "xs:string"/>
```

```
<xs:element name = "author" type= "xs:string"/>
```

```
...
```

```
...
```

```
...
```



# 7.1. Resource Description Framework

- The basic data model used to build the SW is called RDF, a domain-independent model for describing resources
- A resource may be name, place, constant, web pages (URLs), parts of web pages, etc.
- RDF uses URIs to represent data usually in triple structures <subject, relation, object>
- it make statements about web resources as subject-predicate-object expressions, called triples

- Suppose a webpage <http://www.cs.univ.ac.in> has been created by Criss, then RDF terms for the various parts of the statement are:
  - The subject is the URL: <http://www.cs.univ.ac.in>
  - The predicate is the word “creator”
  - The object is the phrase “Criss”
- Corresponding RDF statement is as follows:  
<http://www.cs.univ.ac.in/~criss> <rel: creator> <“Criss”>