

Knowledge Representation

Unit-2

Syllabus

- Propositional logic and predicate logic
- Procedural vs. Declarative knowledge
- Knowledge Representation structure such as frame, Semantic networks and script
- Resolution in predicate logic, forward vs. backward reasoning
- Non-monotonic Reasoning, Logics for non-monotonic reasoning

What is Knowledge?

knowledge, *nol'ij*, n. assured belief; that which is known; information; ...

In order to solve the complex problems encountered in AI, one generally needs a large amount of knowledge, and suitable mechanisms for representing and manipulating all that knowledge.

Knowledge can take many forms. Some simple examples are:

John has an umbrella

It is raining

An umbrella stops you getting wet when it's raining

An umbrella will only stop you getting wet if it is used properly

Umbrellas are not so useful when it is very windy

What is a Knowledge Representation?

The object of a *knowledge representation* is to express knowledge in a computer tractable form, so that it can be used to enable our AI agents to perform well.

A *knowledge representation language* is defined by two aspects:

1. **Syntax** The syntax of a language defines which configurations of the components of the language constitute valid sentences.
2. **Semantics** The semantics defines which facts in the world the sentences refer to, and hence the statement about the world that each sentence makes.

Suppose the language is arithmetic, then

‘ x ’, ‘ \geq ’ and ‘ y ’ are *components* (or symbols or words) of the language

the *syntax* says that ‘ $x \geq y$ ’ is a valid sentence in the language, but ‘ $\geq \geq x y$ ’ is not

the *semantics* say that ‘ $x \geq y$ ’ is false if y is bigger than x , and true otherwise

Requirements of a Knowledge Representation

A good knowledge representation system for any particular domain should possess the following properties:

1. **Representational Adequacy** – the ability to represent all the different kinds of knowledge that might be needed in that domain.
2. **Inferential Adequacy** – the ability to manipulate the representational structures to derive new structures (corresponding to new knowledge) from existing structures.
3. **Inferential Efficiency** – the ability to incorporate additional information into the knowledge structure which can be used to focus the attention of the inference mechanisms in the most promising directions.
4. **Acquisitional Efficiency** – the ability to acquire new information easily. Ideally the agent should be able to control its own knowledge acquisition, but direct insertion of information by a ‘knowledge engineer’ would be acceptable.

Finding a system that optimises these for all possible domains is not going to be feasible.

Techniques of knowledge Representation

1. Logical Representation
2. Semantic Networks
3. Frames
4. Scripts

Logic

“Logical AI:

The idea is that an agent can represent knowledge of its world, its goals and the current situation by sentences in logic and decide what to do by inferring that a certain action or course of action is appropriate to achieve its goals.”

John McCarthy in Concepts of logical AI, 2000.

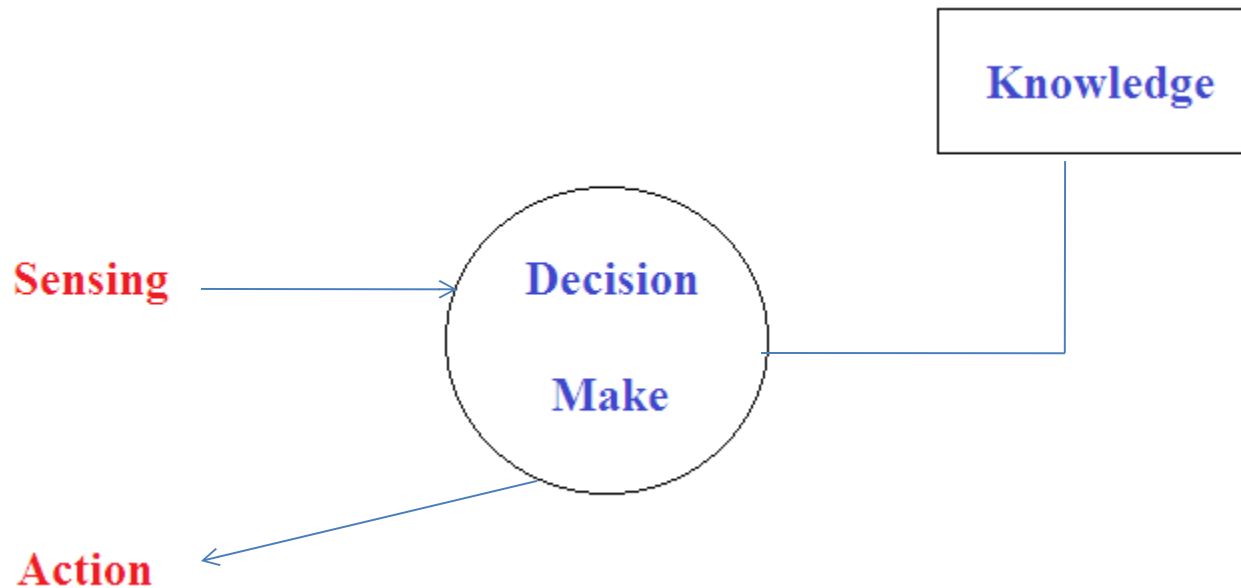
<http://www-formal.stanford.edu/jmc/concepts-ai/concepts-ai.html>

Knowledge Based Agent

- **Intelligent Agent** : Perceiving, Knowledge Representation, Reasoning and Acting
- The primary component of a knowledge-based agent is its **knowledge-base**
- A knowledge-base is a **set of sentences**
- Each sentence is expressed in a language called the **knowledge representation language**.
- There must mechanisms to **derive new sentences from old ones**. This process is known as inferencing or reasoning

Knowledge and Intelligence

- Does knowledge have any role in demonstrating Intelligent behavior.



Knowledge-based agents

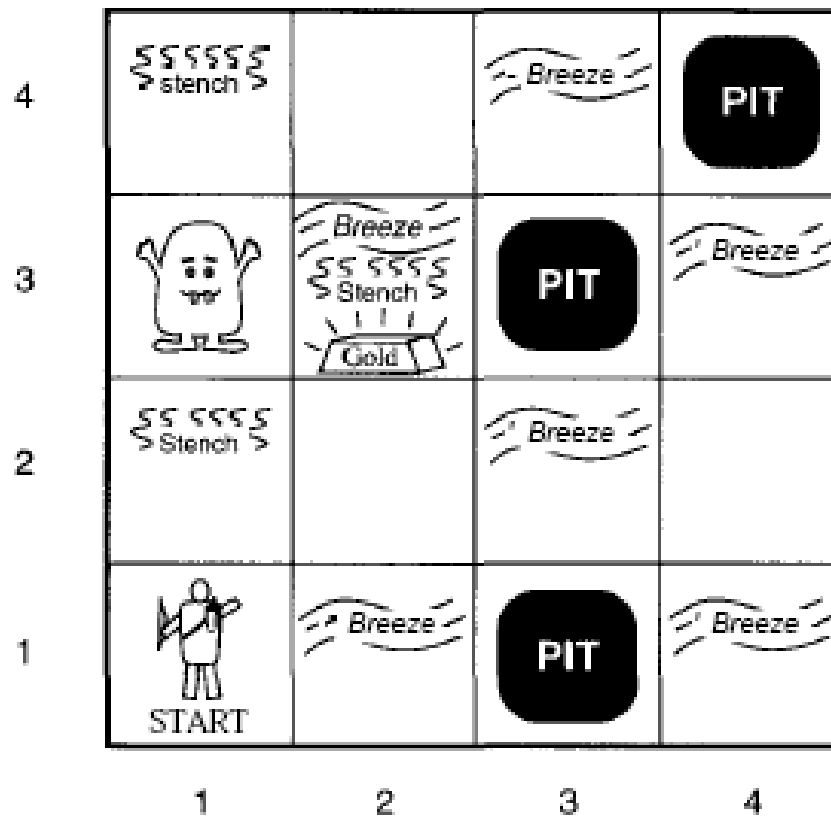
- The agent **must be able to**:
 - Represent states, actions, etc.
 - Incorporate new percepts
 - Update internal representations of the world
 - Deduce hidden properties of the world
 - Deduce appropriate actions
- **Declarative** approach to building an agent:
 - Add new sentences: **Tell** it what it needs to know
 - Query what is known: **Ask** itself what to do - answers should follow from the KB

Knowledge Based Agent

- There must be a way to **add new sentences** to the **knowledge base**, and a way to query what is known
- The standard names for these tasks are **TELL** and **ASK**

```
function KB-AGENT(percept) returns an action  
  static: KB, a knowledge base  
          t, a counter, initially 0, indicating time  
  
  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))  
  action ← ASK(KB, MAKE-ACTION-QUERY(t))  
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))  
  t ← t + 1  
  return action
```

The Wumpus World Environment



The Wumpus World Environment

- Adjacent means left, right, top or bottom
- **Stench** : In squares containing and adjacent to wumpus
- **Breeze** : In squares adjacent to a pit

There can be one wumpus, one gold, and many pits. Agent starts from the bottom left square of a grid.

The Wumpus World Environment

- The agent dies if it enters a square containing a pit or the wumpus
- The agent can shoot the wumpus along a straight line
- The agent has only one arrow

Wumpus World PEAS

- **Performance measure:** gold +1000, death (eaten or falling in a pit) -1000, -1 per action taken, -10 for using the arrow. The game ends either when the agent dies or comes out of the cave.
- **Environment**
 - 4 X 4 grid of rooms
 - Agent starts in square [1,1] facing to the right
 - Locations of the gold, and Wumpus are chosen randomly with a uniform distribution from all squares except [1,1]
 - Each square other than the start can be a pit with probability of 0.2

Wumpus World PEAS

- **Actuators:**
 - Left turn, Right turn, Forward, Grab, Release, Shoot
- **Sensors:**
 - Stench, Breeze, Glitter, Bump, Scream
 - Represented as a 5-element list
 - Example: [Stench, Breeze, None, None, None]

Wumpus World properties

- Partially observable
- Static
- Discrete
- Single-agent
- Deterministic
- Sequential

Exploring Wumpus World

Agent's first steps:

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2	2,2	3,2	4,2
OK			
1,1	2,1	3,1	4,1
A			
OK	OK		

(a)

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2	2,2	3,2	4,2
OK	P?		
1,1	2,1	3,1	4,1
V	A	P?	
OK	B		
	OK		

(b)

Exploring Wumpus World

Agent's later steps:

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(a)

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 A S G B	3,3 P?	4,3
1,2 S V OK	2,2 V OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(b)

Logic

- **Knowledge base**: a set of sentences in a formal representation, **logic**
- **Logics**: are formal languages for representing knowledge to extract conclusions
 - **Syntax**: defines well-formed sentences in the language
 - **Semantic**: defines the truth or meaning of sentences in a world
- **Inference**: a procedure to derive a new sentence from other ones.
- **Logical entailment**: is a relationship between sentences. It means that a sentence **follows logically** from other sentences

$$KB \models \alpha$$

Types of Logic

There are a number of logical systems with different syntax and semantics.

1. Propositional logic
2. Predicate logic

Propositional Logic

- **Simplest form of logic** where all the statements are made by propositions.
- A proposition is a **declarative statement** which is either true or false
- It is a technique of knowledge representation in logical and mathematical form

Examples -

- a) It is Sunday.
- b) The Sun rises from West (False proposition)
- c) $3+3=7$ (False proposition)
- d) 5 is a prime number.

Propositional Logic

- Propositional logic is a **symbolic logic** for manipulating propositions
- Propositional logic is concerned with **declarative sentences** that can be **classified as either true or false**
- Propositional logic is also called **Boolean logic** as it works on 0 and 1.
- Use **symbolic variables to represent the logic**, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.
- **Propositions can be either true or false**, but it cannot be both.

Propositional Logic

- A sentence whose truth value can be determined is called a **statement** or **proposition**
- A statement is also called a **closed sentence** because its truth value is not open to question
- Statements that cannot be answered absolutely are called **open sentences**
- A **compound statement** is formed by using logical connectives on individual statements
- Statements which are questions, commands, or opinions are not propositions such as "**Where is Rohini**", "**How are you**", "**What is your name**", are not propositions.

Syntax of propositional logic

- The syntax of propositional logic defines the allowable sentences for the knowledge representation. There are two types of Propositions:

1.Atomic Propositions

2.Compound propositions

- **Atomic Proposition:** Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.

"The Sun is cold" is also a proposition as it is a **false** fact.

- **Compound proposition:** Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.

"It is raining today, and street is wet."

- In **propositional logic (PL)** an user defines a set of propositional symbols, like ***P*** and ***Q***. User defines the semantics of each of these symbols. For example :
 - P means "It is hot"
 - Q means "It is humid"
 - R means "It is raining"
- A **sentence (also called a formula or well-formed formula or wff)** is defined as:
 1. A symbol
 2. If S is a sentence, then $\sim S$ is a sentence, where " \sim " is the "not" logical operator
 3. If S and T are sentences, then $(S \vee T)$, $(S \wedge T)$, $(S \Rightarrow T)$, and $(S \Leftrightarrow T)$ are sentences, where the four logical connectives correspond to "or," "and," "implies," and "if and only if," respectively

Propositional logic (PL)

For example,

- **P** means "It is hot"
- **Q** means "It is humid"
- **R** means "It is raining"

Examples of PL sentences:

◦ $(P \wedge Q) \Rightarrow R$

(here meaning "If it is hot and humid, then it is raining")

◦ $Q \Rightarrow P$

(here meaning "If it is humid, then it is hot")

◦ Q

(here meaning "It is humid.")

Propositional logic (PL)

- Propositional logic All objects described are fixed or unique

“John is a student” $\text{student}(\text{john})$

Here John refers to one unique person.

Propositional Calculus Sentences

- Every propositional symbol and truth symbol is a *sentence*.

Examples: true, P, Q, R.

- The *negation* of a sentence is a sentence.

Examples: $\neg P$, \neg false.

- The *conjunction*, or *and*, of two sentences is a sentence.

Example: $P \wedge \neg P$

Propositional Calculus Sentences (cont'd)

- The *disjunction*, or *or*, of two sentences is a sentence.

Example: $P \vee \neg P$

- The *implication* of one sentence from another is a sentence.

Example: $P \rightarrow Q$

- The *equivalence* of two sentences is a sentence.

Example: $P \vee Q \equiv R$

- Legal sentences are also called well-formed formulas or *WFFs*.

Properties of Operators:

- **Commutativity:**
 - $P \wedge Q = Q \wedge P$, or
 - $P \vee Q = Q \vee P$.
- **Associativity:**
 - $(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$,
 - $(P \vee Q) \vee R = P \vee (Q \vee R)$
- **Identity element:**
 - $P \wedge \text{True} = P$,
 - $P \vee \text{True} = \text{True}$

Properties of Operators:

- **Distributive:**
 - $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R).$
 - $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R).$
- **DE Morgan's Law:**
 - $\neg (P \wedge Q) = (\neg P) \vee (\neg Q)$
 - $\neg (P \vee Q) = (\neg P) \wedge (\neg Q).$
- **Double-negation elimination:**
 - $\neg (\neg P) = P.$

Objectives

- Infer truth value of proposition
- Reason towards new facts, given set of propositions

Inference In Propositional Logic

Procedure to derive Truth Value

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

A	B	$A \rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

1. Modus Ponens

The Modus Ponens rule is one of the most important rules of inference, and it states that **if P and $P \rightarrow Q$ is true, then we can infer that Q will be true.** It can be represented as:

Notation for Modus ponens:
$$\frac{P \rightarrow Q, P}{\therefore Q}$$

Example:

Statement-1: "If I am sleepy then I go to bed" $\Rightarrow P \rightarrow Q$


Statement-2: "I am sleepy" $\Rightarrow P$

Conclusion: "I go to bed." $\Rightarrow Q$.

Hence, we can say that, if $P \rightarrow Q$ is true and P is true then Q will be true.

Proof by Truth table:

P	Q	$P \rightarrow Q$
0	0	0
0	1	1
1	0	0
1	1	1



2. Modus Tollens

- The Modus Tollens rule state that **if $P \rightarrow Q$ is true and $\neg Q$ is true, then $\neg P$ will also true.** It can be represented as

Notation for Modus Tollens:
$$\frac{P \rightarrow Q, \neg Q}{\neg P}$$


Statement-1: "If I am sleepy then I go to bed" $\Rightarrow P \rightarrow Q$

Statement-2: "I do not go to the bed." $\Rightarrow \sim Q$

Statement-3: Which infers that "**I am not sleepy**" $\Rightarrow \sim P$

Proof by Truth table:

P	Q	$\sim P$	$\sim Q$	$P \rightarrow Q$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1



Limitation

- We cannot represent relations like **ALL, some, or none** with propositional logic.
Example:
 - **All the girls are intelligent.**
 - **Some apples are sweet.**
- Propositional logic has **limited expressive power.**
- In propositional logic, **we cannot describe statements in terms of their properties or logical relationship**

First Order Logic
OR
Predicate Logic

Limitation of Proposition Logic

- Consider following arguments
 - All dogs are faithful
 - Tommy is a dog
 - Therefore, tommy is faithful

How to represent and infer this in propositional logic ?

Limitation of Proposition Logic

- P is all dogs are faithful
- q is Tommy is a dog
- But $p \wedge q \Rightarrow$ Tommy is faithful ??
- **No ! We can not infer this in proposition Logic**

More Scenarios

- Tom is a hardworking : **Hardworking (Tom)**
 - Tom is intelligent : **Intelligent (Tom)**
 - **Rule** : If Tom is hardworking and intelligent then tom Scores high marks
 - **Hardworking [Tom] ^ Intelligent [Tom] = > Scores_High_Marks[Tom]**
- What about john and Jill

The Problem of Infinite Model

- In general, Propositional logic can deal with only a finite numbers of propositions.
- If there are only three dogs Tommy, Jimmy and Leela the

T : Tommy is faithful

J : Jimmy is faithful

L : Leela is Faithful

All dogs are faithful : $T \wedge J \wedge L$

First Order Logic

- Another way of knowledge representation
- It is an extension to propositional logic
- Predicate Logic deals with predicates, which are propositions, consist of variables
- Also known as Predicate logic
- Powerful language that develops information about the objects

Syntax for FOL

- Defines in terms of
 - Terms
 - Predicates
 - Quantifiers

Terms

- A term denotes some objects other than true or false

➤ **Tommy** is a dog

Terms

➤ All **Men** are mortal

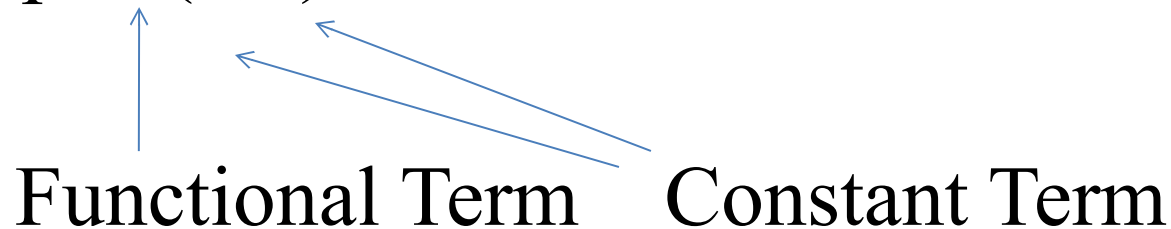
- **Terms** : Constants and Variables

Terms

- A **constant** of type W is a name that denotes a particular objects in a set W
 - **Example** : 5, Tommy
- A **variable** of type W is a name that can denote any element in the set W
- **Example** : $x \in N$ denotes a natural number
 $d \in M$ denotes the name of dog

Terms: Functions

- A functional terms takes **n** objects of type **W1** to **Wn** as input and returns an objects of type **W**.
- $F = (W1, W2, \dots, Wn)$
- $\text{plus}(3, 4) = 7$



Functional Term Constant Term

The diagram illustrates the classification of terms in the expression $\text{plus}(3, 4) = 7$. A vertical blue arrow points from the text 'Functional Term' to the function name 'plus'. Two blue arrows point from the text 'Constant Term' to the arguments '3' and '4'.

Predicates

- Predicate **are like functions** except that their return type is true or false.
- **Example** : $\text{greater}(x, y)$ is true if $x > y$
- Here greater is a predicate symbol that takes two arguments of type natural number
- $\text{greater}[2,4]$ – not valid
- $\text{greater}[3,-4]$ – valid

Types of Predicates

- A predicates with **no variable is a proposition**
 - Tommy is a dog
- A predicate with one variable is called a **property**
 - $\text{dog } [x]$ is true if x is a dog
 - $\text{mortal } [y]$ is true if y is mortal

Formulation of Predicates

- Let $P(x,y,\dots)$ and $Q(x,y,\dots)$ are two predicates
- Then so are
 - $P \vee Q$
 - $P \wedge Q$
 - $\sim P$
 - $P \Rightarrow Q$

Predicates Examples

- If x is a man then x is mortal

$$\mathbf{man(x) \Rightarrow mortal (x)}$$

- If n is a natural number, then n is either even or odd

$$\mathbf{natural(n) = > even (n) \vee odd(n)}$$

Quantifiers

- There are two basic quantifiers in FOL
 - \forall “For all” – Universal quantifier
 - \exists “There Exists” – Existential Quantifiers

- **Universal Quantifiers :**

All dogs are faithful

faithful (x) : x is faithful

dog(x) : x is a dog

$\forall x (\text{dog}(x) \Rightarrow \text{faithful}(x))$

- **All brides cannot fly**

fly (x) : x can fly

birds(x) : x is a bird

$\sim [\forall x (\text{bird } (x) \Rightarrow \text{fly } (x))]$

- **Existential Quantifiers**

➤ **All birds can not fly : there exists a bird that can not fly**

$\text{fly}(x) : x \text{ can fly}$

$\text{birds}(x) : x \text{ is a bird}$

$\exists x [\text{bird}(x) \wedge \sim \text{fly}(x)]$

Inference in First-Order Logic

- Used to deduce new facts or sentences from existing sentences.
- Some basic terminologies used in FOL

Inference in First-Order Logic

1. Substitution –

- fundamental operation performed on terms and formulas.
- It occurs in all inference systems
- The substitution is complex in the presence of quantifiers in FOL.
- $F[a/x]$, so it refers to substitute a constant "a" in place of variable "x".

Inference in First-Order Logic

2. Equality

- First-Order logic does not only use predicate and terms for making atomic sentences but also uses another way, **which is equality in FOL.**
- we can use **equality symbols** which specify that the two terms refer to the same object.
- **Example - Brother (John) = Smith.**
 - Also be used with negation to represent **that two terms are not the same objects.**
- **Example - $\neg (x=y)$ which is equivalent to $x \neq y$.**

FOL inference rules for quantifier

- **Universal Generalization**
- **Universal Instantiation**
- **Existential Instantiation**
- **Existential introduction**

Universal Generalization

- Universal generalization is a **valid inference rule** which states that **if premise $P(c)$ is true** for any arbitrary element c in the universe of discourse, then we can have a **conclusion as $\forall x P(x)$** .
- It can be represented as
$$\frac{P(c)}{\forall x P(x)}$$
- This rule can be used if we want to **show that every element has a similar property**.

Universal Generalization

Example –

➤ $P(c)$: "A byte contains 8 bits"

➤ so for $\forall x P(x)$ "All bytes contain 8 bits.",
it will also be true.

Universal Instantiation

- Also called as **universal elimination**
- It can be applied multiple times **to add new sentences.**
- It can be represented as
$$\frac{\forall x P(x)}{P(c)}$$

Example –

Every person like ice-cream $\Rightarrow \forall x P(x)$

John likes ice-cream $\Rightarrow P(c)$

Existential Instantiation

- Also called as Existential Elimination
- It can be applied only **once to replace the existential sentence.**
- This rule states that one can infer $P(c)$ from the formula given in the form of $\exists x P(x)$ **for a new constant symbol c .**
- It can be represented as
$$\frac{\exists x P(x)}{P(c)}$$

Existential Instantiation

Example –

$\exists x \text{Crown}(x) \wedge \text{OnHead}(x, \text{John})$

So we can infer: $\text{Crown}(K) \wedge \text{OnHead}(K, \text{John})$, as long as K does not appear in the knowledge base.

- The above used K is a constant symbol, which is called **Skolem constant**.
- The Existential instantiation is a special case of **Skolemization process**.

Existential introduction

- Also known as an **existential generalization**
- This rule states that if there is **some element c** in the universe of discourse **which has a property P**, then we can infer that there **exists something in the universe which has the property P**.
- It can be represented as
$$\frac{P(c)}{\exists x P(x)}$$
- **Example: Let's say that,**
"Priyanka got good marks in English."
"Therefore, someone got good marks in English."

Examples

- Some dogs bark
- All dogs have four legs
- All barking dogs are irritating
- No dogs purr
- Fathers are male parents with children
- Students are people who are enrolled in courses

Examples

- Some dogs bark

$$\exists x [\text{dog}(x) \wedge \text{bark}(x)]$$

- All dogs have four legs

$$\forall x [\text{dog}(x) \Rightarrow \text{have_four_leg}(x)]$$

$$\forall x [\text{dog}(x) \Rightarrow \text{legs}(x,4)]$$

Examples

- No dogs purr

$$\neg \exists x [\text{dog}(x) \wedge \text{purr}(x)]$$

- Fathers are male parents has children

$$\forall x [\text{father}(x) \Rightarrow \text{male}(x) \wedge \text{has_children}(x)]$$

- Students are people who are enrolled in courses

$$\forall x [\text{student}(x) \Rightarrow \text{people}(x) \wedge \text{enrolled_course}(x)]$$

Semantic Net

Semantic Networks

- Alternative of predicate logic for knowledge representation.
- Represent our knowledge in the form of graphical networks
- This network consists of nodes representing objects and arcs which describe the relationship between those objects.
- Categorize the object in different forms and can also link those objects.
- Easy to understand and can be easily extended.

Semantic Networks

- This representation consist of mainly two types of relations:
 - 1.IS-A relation (Inheritance)
 - 2.Kind-of-relation

Semantic Networks

- Statements:

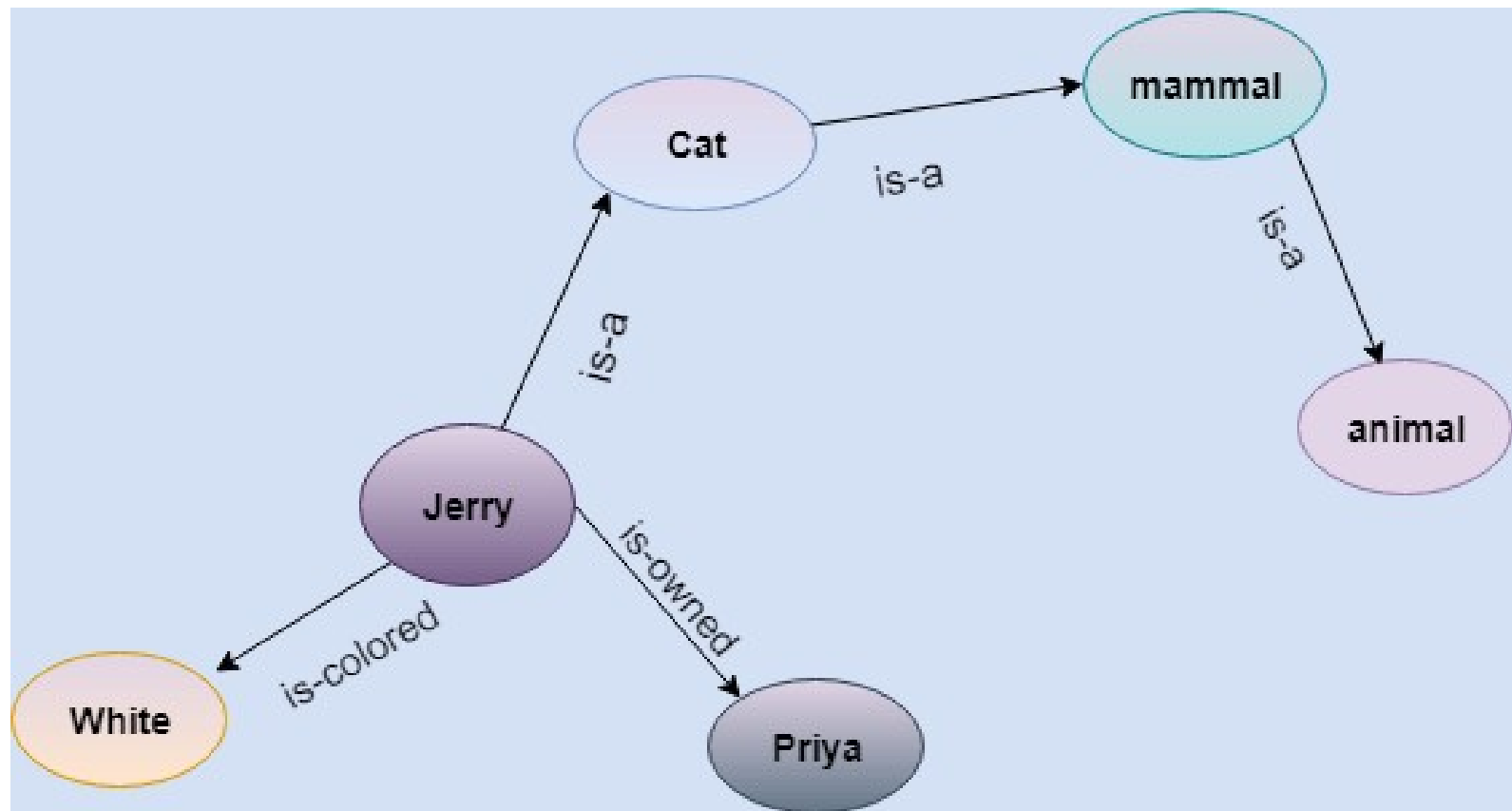
1. Jerry is a cat.

2. Jerry is a mammal.

3. Jerry is owned by Priya.

4. Jerry is white colored.

5. All Mammals are animal.



Drawbacks

- More computational **time**
- In practice, **it is not possible to build** such a vast semantic network.
- These types of representations are **inadequate as they do not have any equivalent quantifier**, e.g., for all, for some, none, etc.
- Semantic networks do not have any **standard definition for the link names**.
- These networks are **not intelligent** and depend on the creator of the system.

Advantages

1. Semantic networks are a natural representation of knowledge.
2. Semantic networks convey meaning in a transparent manner.
3. These networks are simple and easily understandable.

Frames

Frame

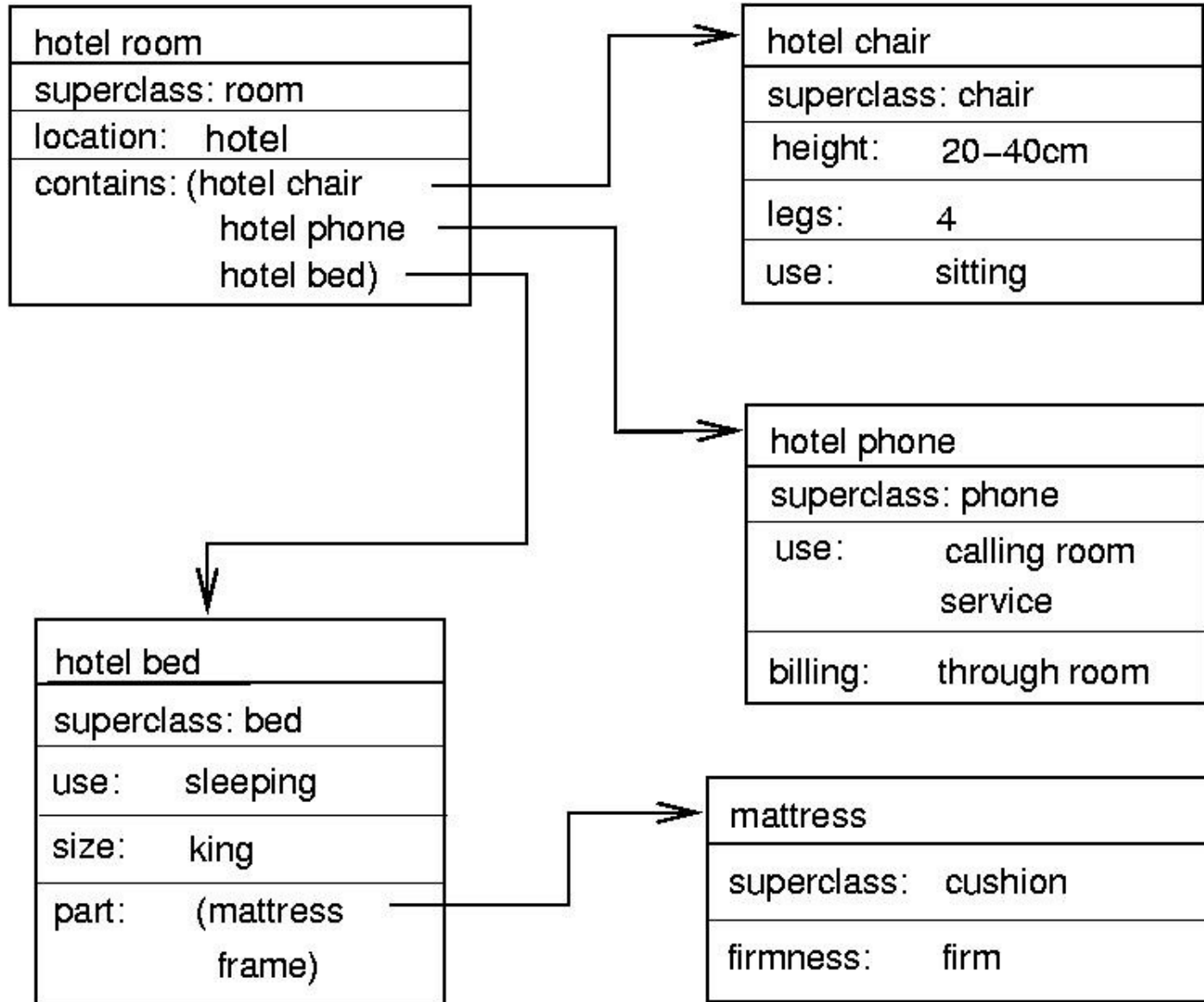
- By Marvin Minsky in 1970
- **Definition-** A collection of **attributes** and associated **values** that describe **some entity** in the world
- A frame represents a concept
- Frames are organized **into hierarchies or network of frames**
- Lower-level frames can **inherit information from upper-level frames** in network

Frame

- A Frame system is a **collection of objects**.
- Each object contains a **number of slots**.
- A **slot represents an attribute**. Each slot has a **value**. The value of an **attribute can be another object**.
- Each object is like a C struct. The struct has a name and contains a bunch of named values (which can be pointers)

Frame

- A Frame system is a **collection of objects**.
- Each object contains a **number of slots**.
- A **slot represents an attribute**. Each slot has a **value**. The value of an **attribute can be another object**.
- Each object is like a C struct. The struct has a name and contains a bunch of named values (which can be pointers)



Advantages

- **Allowing complex object** represented as a single frame
- **Provide easier framework** to organize the information hierarchically
- Frame support **class inheritance**

Script

- Another knowledge representation technique
- Frame like structures used to represent commonly occurring experiences (going to restaurant, visiting a doctor)
- It is a structure that describes a stereotyped sequence of events in a particular context.
- A script consist of a set of slots.
- Associated with each slot may be some information

Script

- Useful because in the real world, there are **no patterns to the occurrence of events**.
- Patterns arise because of **clausal relationships** between events.
- The events described in a script form **a giant casual chain**
- The beginning of the chain is the set **of entry conditions**
- The end of the chain is the **set of results**

Script

A script has various components like:

- **Entry condition:** It must be true before the events described in the script can occur
- **Tracks:** It specifies particular position of the script
- **Result:** It must be satisfied or true after the events described in the script have occurred.
- **Probs:** It describes the inactive or dead participants in the script
- **Roles:** It specifies the various stages of the script.
- **Scenes :** The sequence of events that occur.

- Describing a script, special symbols of actions are used. These are:

Symbol	Meaning	Example
ATRANS	transfer a relationship	give
PTRANS	transfer physical location of an object	go
PROPEL	apply physical force to an object	push
MOVE	move body part by owner	kick
GRASP	grab an object by an actor	hold
INGEST	taking an object by an animal eat	drink
EXPEL	expel from animal's body	cry
MTRANS	transfer mental information	tell
MBUILD	mentally make new information	decide
CONC	conceptualize or think about an idea	think
SPEAK	produce sound	say
ATTEND	focus sense organ	listen

Script Example

- a) **Script name** : Movie
- b) **Track** : CINEMA HALL
- c) **Roles** : Customer(C), Ticket seller(TS), Ticket Checker(TC), Snacks Sellers (SS)
- d) **Probes** : Ticket, snacks, chair, money, Ticket, chart
- e) **Entry condition** : The customer has money
The customer has interest to watch movie.
- f) **Scenes** : The sequence of events that occur.

SCENE-1 (Entering into the cinema hall)

- ✓ C PTRANS C into the cinema hall
- ✓ C ATTEND eyes towards the ticket counter
- ✓ C PTRANS C towards the ticket counters
- ✓ C ATTEND eyes to the ticket chart
- ✓ C MBUILD to take which class ticket C
MTRANS TS for ticket
- ✓ C ATRANS money to TS
- ✓ TS ATRANS ticket to C

Script

- **Advantages of Scripts**

- Ability to **predict events**.
- A single **coherent interpretation** maybe builds up from a collection of observations.

- **Disadvantages of Scripts**

- **Less general** than frames.
- **May not be suitable** to represent all kinds of knowledge

Sematic Network Example

Tom is an instance of dog.

Tom caught a cat

Tom is owned by rashan.

Tom is brown in colour.

Dogs like bones.

The dog sat on the mat.

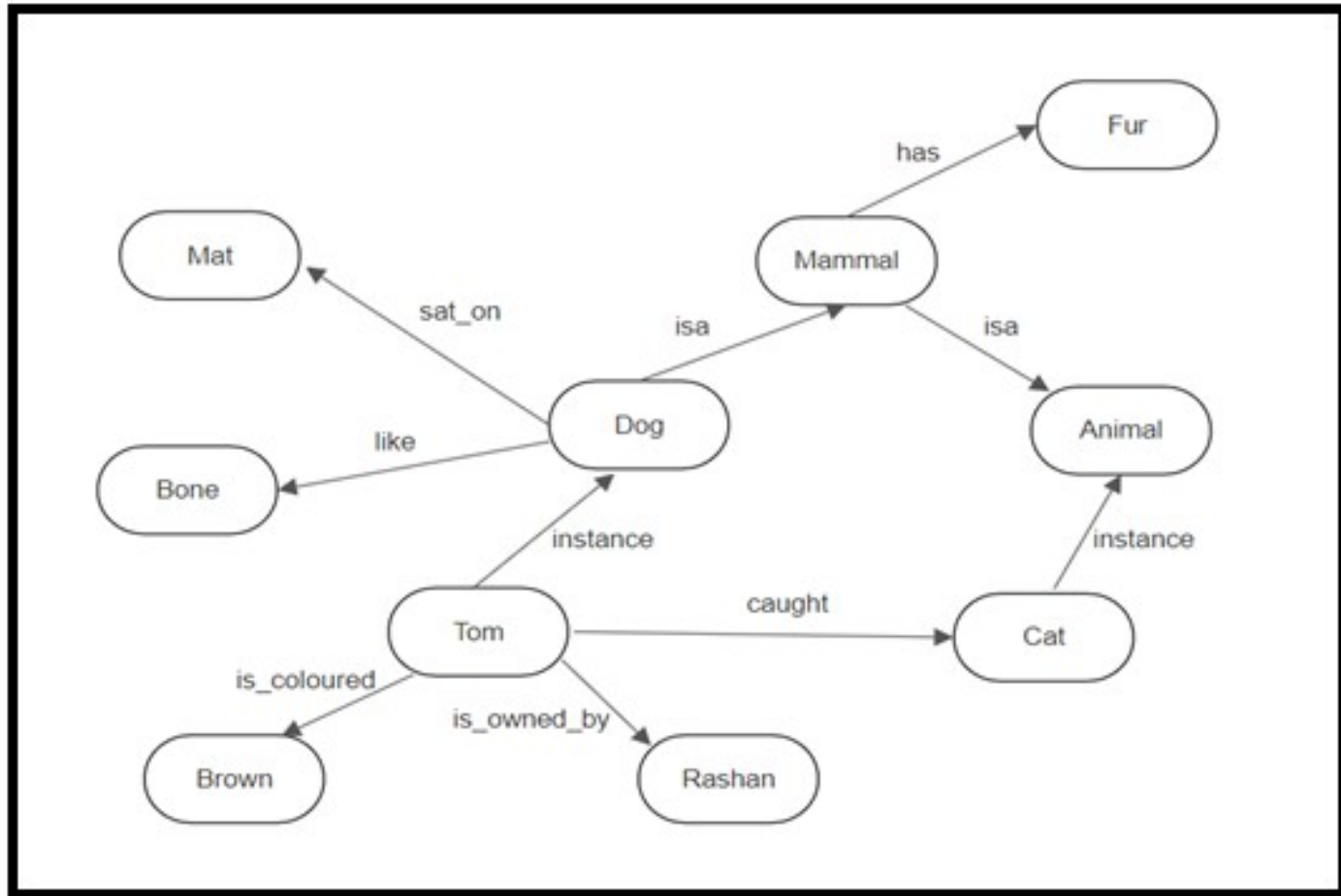
A dog is a mammal.

A cat is an instance animal

All mammals are animals.

Mammals have fur.

Semantic Network Example



Solving Problems

- Convert following first order predicate into clausal normal form
 1. All people who are graduating are happy
 2. All happy people smile
 3. Someone is graduating

1. Every gardener likes the sun.
2. All purple mushrooms are poisonous
3. No purple mushroom is poisonous

Predicates

- $\forall [x] [\text{graduating } [x] \rightarrow \text{happy } [x]]$
- $\forall [x] [\text{happy } [x] \rightarrow \text{smiling}[x]]$
- $\exists [x] [\text{graduating } [x]]$

Predicates

1. Every gardener likes the sun.

$$(\forall x) (\text{gardener}(x) \rightarrow \text{likes}(x, \text{sun}))$$

2. All purple mushrooms are poisonous

$$(\forall x) [(\text{mushrooms}(x) \wedge \text{purple}(x)) \rightarrow \text{poisonous}(x)]$$

3. No purple mushroom is poisonous

$$\sim(\exists x) (\text{purple}(x) \wedge \text{mushroom}(x) \wedge \text{poisonous}(x))$$

- **Convert the following English statements to statements in first order logic**
 - a. every boy or girl is a child
 - b. every child gets a doll or a train or a lump of coal
 - c. no boy gets any doll
 - d. no child who is good gets any lump of coal
 - e. Jack is a boy

- **Write a Script for going to the bank to withdraw money.**

Script for going to the bank to withdraw money.

- **SCRIPT** : Withdraw money
- **TRACK** : Bank
- **PROPS** : Money, Counter, Form, Token
- **Roles** : P= Customer
- E= Employee
- C= Cashier
- **Entry conditions**: P has no or less money.
- The bank is open.
- **Results** : P has more money.

- **SCRIPT : Withdraw money**
- **TRACK : Bank**
- **PROPS : Money, Counter, Form, Token**
- **Roles : P= Customer, E= Employee**
C= Cashier
- **Entry conditions: P has no or less money.**
The bank is open.
- **Results : P has more money.**
- **Scene 1: Entering**
- **P PTRANS P into the Bank**
- **P ATTEND eyes to E**
- **P MOVE P to E**

- **The FOL statements are**

a. forall x ((boy(x) or girl(x)) -> child(x))

b. forall y (child(y) -> (gets(y,doll) or gets(y,train)
or gets(y,coal)))

c. forall w (boy(w) -> !gets(w,doll))

d. forall z ((child(z) and good(z)) -> !gets(z,coal))

e. boy(Jack)

Forward and Backward Chaining

Inference engine

- It is a component of the intelligent system
 - Which applies logical rules to the knowledge base to **infer new information** from known facts
 - The first inference engine was part of the expert system
 - Commonly proceeds in two modes
- 1. Forward chaining**
 - 2. Backward chaining**

Forward Chaining

- It is also known as a **forward deduction or forward reasoning** method
- It is a form of reasoning which **start with atomic sentences** in the knowledge base and applies **inference rules** (Modus Ponens) in the forward direction to extract more data until a goal is reached.
- Starts from **known facts**, triggers all rules whose premises are satisfied, and add their conclusion to the known facts. This process repeats until the problem is solved.

Properties of Forward-Chaining

- Down-up approach
- Making a **conclusion** based on known facts or data
- **Data-driven** as we reach to the goal using available data.
- Commonly used in the **expert system**, such as CLIPS, business, and production rule systems.

Example Knowledge Base

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy America, has some missiles, and all of its missiles were sold to it by Colonel West, who is an American.
- **Prove that Colonel West is a criminal.**

Example Knowledge Base

It is a crime for an American to sell weapons to hostile nations

American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)

The country Nono, an enemy America

Enemy(nono, America)

Nono...has some missiles

Owns(Nono, M1)

Missile(M1)

All of its missiles were sold to it by Colonel. West

$\forall x$ Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(Colonel, x, Nono)

Example Knowledge Base

Missiles are weapons

Missile(x) \Rightarrow Weapon(x)

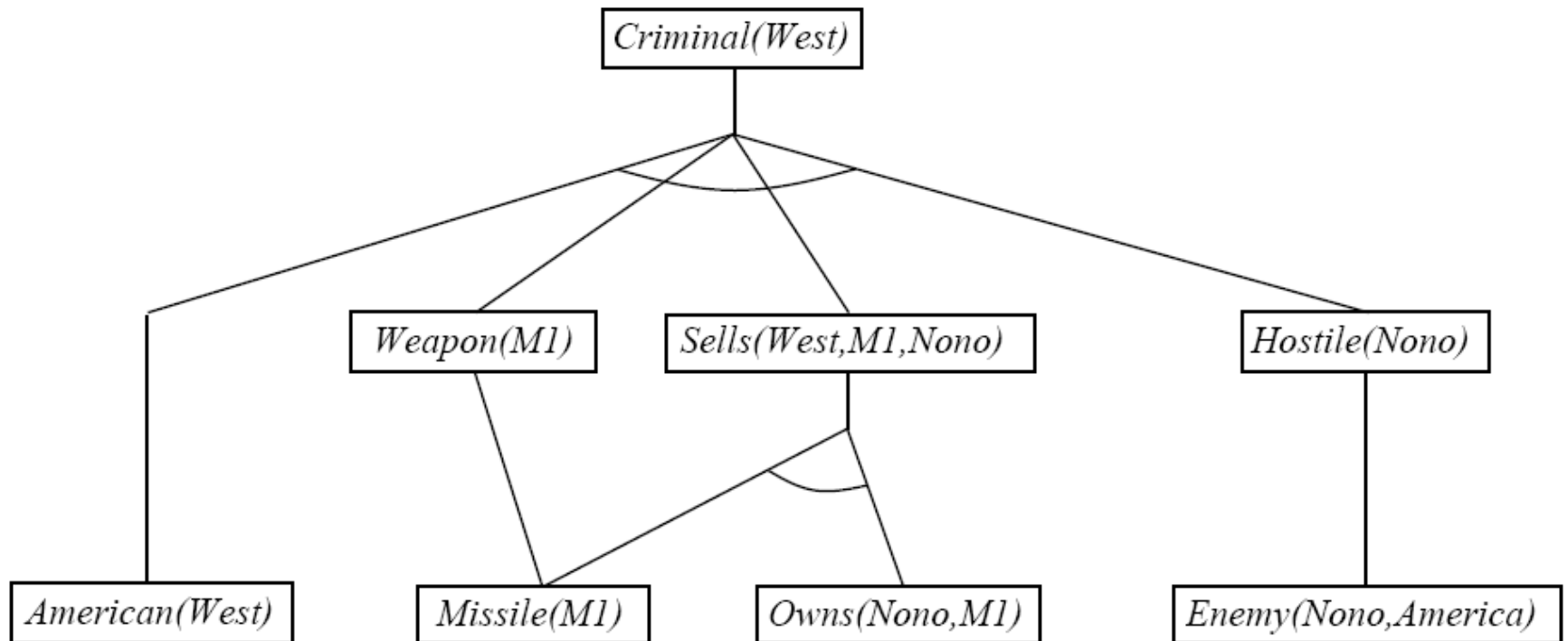
Col. West who is an American

American(Col. West)

An enemy of America counts as “hostile”

Enemy(x, America) \Rightarrow Hostile(x)

Example Knowledge Base

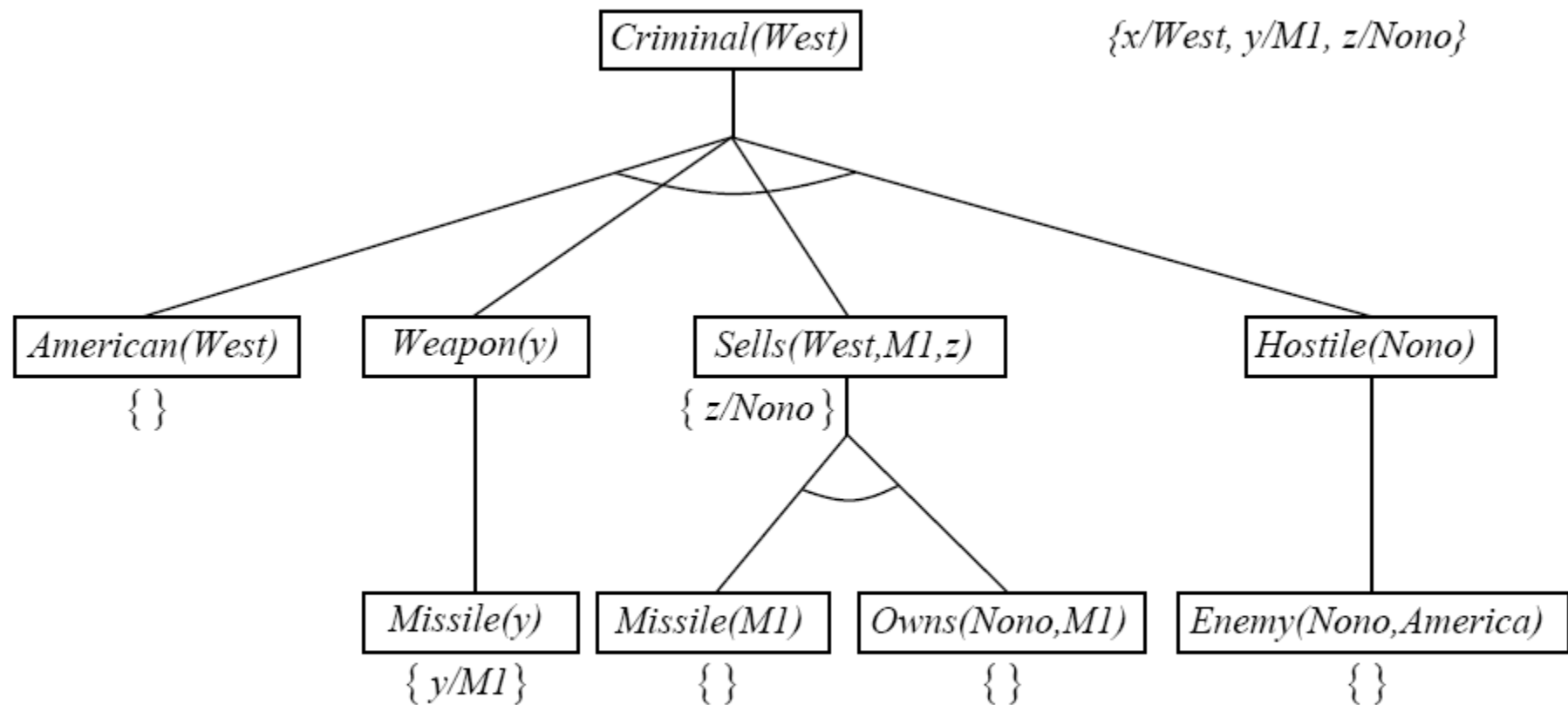


Backward Chaining

- It is also known as a backward deduction or backward reasoning method
- It is a form of reasoning, which starts with the goal and works backward
- Chaining through rules to find known facts that support the goal.

Properties of backward chaining

- Top-down approach.
- Based on modus ponens inference rule
- The goal is broken into sub-goal or sub-goals to prove the facts true.
- Goal-driven approach, as a list of goals decides which rules are selected and used.
- Used in game theory, automated theorem proving tools, inference engines, proof assistants
- mostly used a **depth-first search** strategy for proof.



Resolution in predicate logic

- Theorem proving technique that proceeds by building refutation proofs i.e., **proofs by contradictions.**
- Used, if there are various statements are given, and we **need to prove a conclusion of those statements.**
- Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**

Resolution in predicate logic

- **Clause:** Disjunction of literals (an atomic sentence) is called a **clause**. It is also known as a unit clause.
- **Conjunctive Normal Form:** A sentence represented as a conjunction of clauses is said to be **conjunctive normal form** or **CNF**.

Steps for Resolution

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove
(proof by contradiction)
4. Draw resolution graph (unification).

Example

1. John likes all kind of food.
2. Apple and vegetable are food
3. Anything anyone eats and not killed is food.
4. Anil eats peanuts and still alive
5. Harry eats everything that Anil eats.

Prove by resolution that:

John likes peanuts.

Step-1: Conversion of Facts into FOL

- a. $\forall x: \text{food}(x) \rightarrow \text{likes}(\text{John}, x)$
 - b. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
 - c. $\forall x \forall y: \text{eats}(x, y) \wedge \neg \text{killed}(x) \rightarrow \text{food}(y)$
 - d. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$.
 - e. $\forall x: \text{eats}(\text{Anil}, x) \rightarrow \text{eats}(\text{Harry}, x)$
 - f. $\forall x: \neg \text{killed}(x) \rightarrow \text{alive}(x)$
 - g. $\forall x: \text{alive}(x) \rightarrow \neg \text{killed}(x)$
 - h. $\text{likes}(\text{John}, \text{Peanuts})$
- } **added predicates.**

Step-2: Conversion of FOL into CNF

- **Eliminate all implication (\rightarrow) and rewrite**

1. $\forall x \neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
2. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
3. $\forall x \forall y \neg [\text{eats}(x, y) \wedge \neg \text{killed}(x)] \vee \text{food}(y)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
5. $\forall x \neg \text{eats}(\text{Anil}, x) \vee \text{eats}(\text{Harry}, x)$
6. $\forall x \neg [\neg \text{killed}(x)] \vee \text{alive}(x)$
7. $\forall x \neg \text{alive}(x) \vee \neg \text{killed}(x)$
8. $\text{likes}(\text{John}, \text{Peanuts})$.

Step-2: Conversion of FOL into CNF

Move negation (\neg) inwards and rewrite

1. $\forall x \neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
2. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
3. $\forall x \forall y \neg \text{eats}(x, y) \vee \text{killed}(x) \vee \text{food}(y)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
5. $\forall x \neg \text{eats}(\text{Anil}, x) \vee \text{eats}(\text{Harry}, x)$
6. $\forall x \text{killed}(x) \vee \text{alive}(x)$
7. $\forall x \neg \text{alive}(x) \vee \neg \text{killed}(x)$
8. $\text{likes}(\text{John}, \text{Peanuts}).$

- **Rename variables or standardize variables**

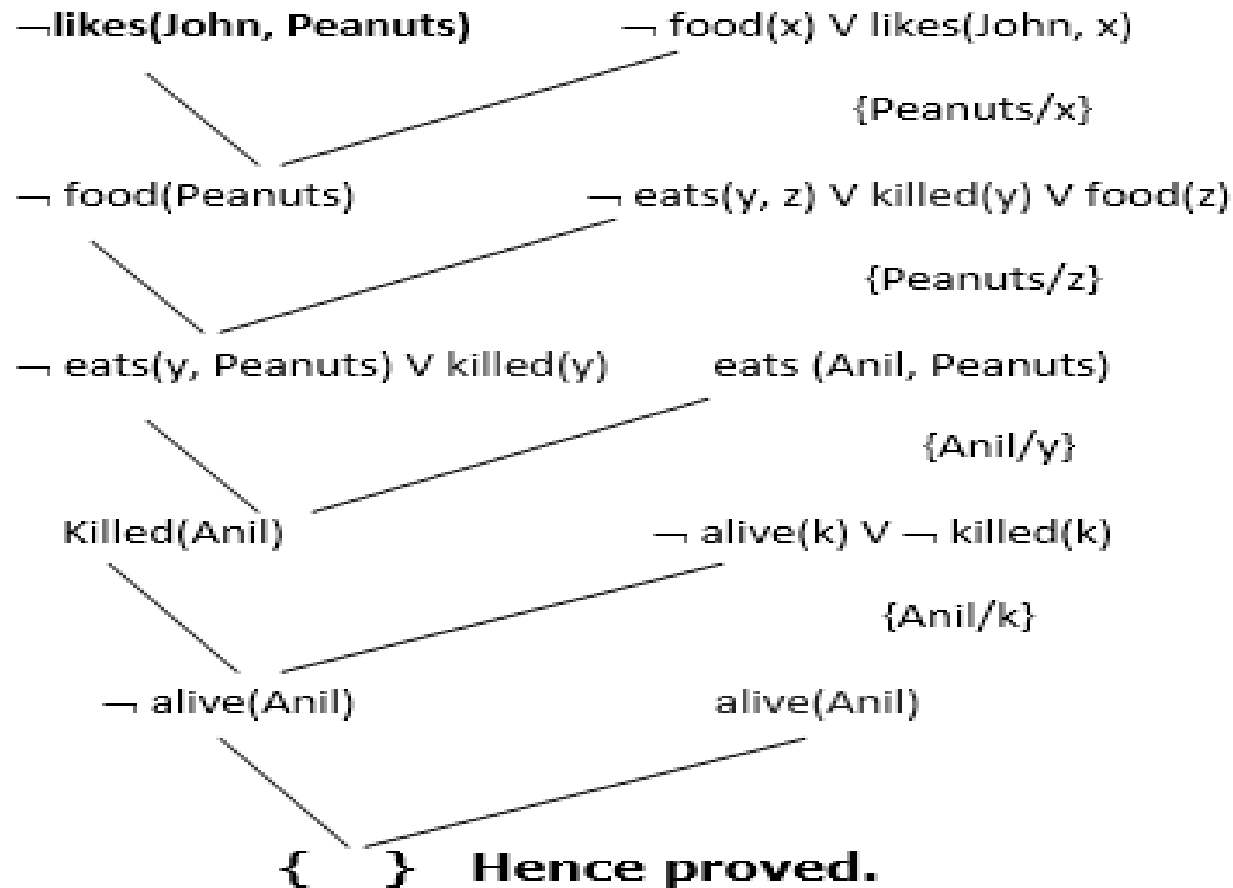
1. $\forall x \neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
2. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
3. $\forall y \forall z \neg \text{eats}(y, z) \vee \text{killed}(y) \vee \text{food}(z)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
5. $\forall w \neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$
6. $\forall g \neg \text{killed}(g) \vee \text{alive}(g)$
7. $\forall k \neg \text{alive}(k) \vee \neg \text{killed}(k)$
8. $\text{likes}(\text{John}, \text{Peanuts})$.

- **Eliminate existential instantiation quantifier by elimination**
- **Drop Universal quantifiers**
 1. $\text{food}(x) \vee \text{likes}(\text{John}, x)$
 2. $\text{food}(\text{Apple})$
 3. $\text{food}(\text{vegetables})$
 4. $\neg \text{eats}(y, z) \vee \text{killed}(y) \vee \text{food}(z)$
 5. $\text{eats}(\text{Anil}, \text{Peanuts})$
 6. $\text{alive}(\text{Anil})$
 7. $\neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$
 8. $\text{killed}(g) \vee \text{alive}(g)$
 9. $\neg \text{alive}(k) \vee \neg \text{killed}(k)$
 10. $\text{likes}(\text{John}, \text{Peanuts})$.

Step-3: Negate the statement to be proved

$\neg \text{likes}(\text{John}, \text{Peanuts})$

Step-4: Draw Resolution graph



Reasoning in Artificial intelligence

- The reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts, and beliefs.
- Reasoning is a way to infer facts from existing data.
- It is a general process of thinking rationally, to find valid conclusions.

Monotonic Reasoning

- In monotonic reasoning, once the **conclusion is taken**, then it will remain the same even if we **add some other information** to existing information in our knowledge base.
- Adding knowledge **does not decrease the set of prepositions that can be derived**.
- Monotonic reasoning is not useful for **the real-time systems**, as in real time, facts get changed, so we cannot use monotonic reasoning.

Monotonic Reasoning

- Monotonic reasoning is used in conventional reasoning systems, and a logic-based system is monotonic.
- Any theorem proving is an example of monotonic reasoning.
- **Example -**

Earth revolves around the Sun.

Advantages of Monotonic Reasoning

- In monotonic reasoning, each old proof will always remain valid.
- If we deduce some facts from available facts, then it will remain valid for always.

Disadvantages of Monotonic Reasoning

- We cannot represent the real-world scenarios using Monotonic reasoning.
- Hypothesis knowledge cannot be expressed with monotonic reasoning, which means facts should be true.
- Since we can only derive conclusions from the old proofs, so new knowledge from the real world cannot be added.

Non-monotonic Reasoning

- In Non-monotonic reasoning, **some conclusions may be invalidated** if we add some more information to our knowledge base.
- **Logic will be said as non-monotonic** if some conclusions can be invalidated by adding more knowledge into our knowledge base.
- Non-monotonic reasoning **deals with incomplete and uncertain models.**

Non-monotonic Reasoning

Example –

Knowledge Base

- ✓ **Birds can fly**
 - ✓ **Penguins cannot fly**
 - ✓ **Pitty is a bird**
- ✓ So from the above sentences, we can conclude that **Pitty can fly.**

Non-monotonic Reasoning

Example –

Knowledge Base

- ✓ Birds can fly
 - ✓ Penguins cannot fly
 - ✓ Pitty is a bird
 - ✓ Pitty is a penguin (Newly Added Knowledge)
- ✓ which concludes "Pitty cannot fly", so it invalidates the above conclusion.

Advantages of Non-monotonic Reasoning

- For real-world systems such as **Robot navigation**, we can use non-monotonic reasoning.
- In Non-monotonic reasoning, **we can choose probabilistic facts or can make assumptions.**

Disadvantages of Non-monotonic Reasoning

- In non-monotonic reasoning, the old facts may be invalidated by adding new sentences.
- It cannot be used for theorem proving.

Resolution in predicate logic

- Theorem proving technique that proceeds by building refutation proofs i.e., **proofs by contradictions.**
- Used, if there are various statements are given, and we **need to prove a conclusion of those statements.**
- Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**

Resolution in predicate logic

- **Clause:** Disjunction of literals (an atomic sentence) is called a **clause**. It is also known as a unit clause.
- **Conjunctive Normal Form:** A sentence represented as a conjunction of clauses is said to be **conjunctive normal form** or **CNF**.

[Animal (g(x) \vee Loves (f(x), x)] and
[\neg Loves(a, b) \vee \neg Kills(a, b)]

- Where two complimentary literals are:
Loves (f(x), x) and \neg Loves (a, b)
- These literals can be unified with unifier $\theta =$
[a/f(x), and b/x], and it will generate a
resolvent clause:

[Animal (g(x) \vee \neg Kills(f(x), x)].

Steps for Resolution

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove
(proof by contradiction)
4. Draw resolution graph (unification).

Example

1. John likes all kind of food.
2. Apple and vegetable are food
3. Anything anyone eats and not killed is food.
4. Anil eats peanuts and still alive
5. Harry eats everything that Anil eats.

Prove by resolution that:

John likes peanuts.

Step-1: Conversion of Facts into FOL

- a. $\forall x: \text{food}(x) \rightarrow \text{likes}(\text{John}, x)$
 - b. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
 - c. $\forall x \forall y: \text{eats}(x, y) \wedge \neg \text{killed}(x) \rightarrow \text{food}(y)$
 - d. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
 - e. $\forall x: \text{eats}(\text{Anil}, x) \rightarrow \text{eats}(\text{Harry}, x)$
 - f. $\forall x: \neg \text{killed}(x) \rightarrow \text{alive}(x)$
 - g. $\forall x: \text{alive}(x) \rightarrow \neg \text{killed}(x)$
 - h. $\text{likes}(\text{John}, \text{Peanuts})$
- } added predicates.

1. John likes all kind of food.
2. Apple and vegetable are food
3. Anything anyone eats and not killed is food.
4. Anil eats peanuts and still alive
5. Harry eats everything that Anil eats.

Prove by resolution that:

John likes peanuts.

Step-2: Conversion of FOL into CNF

- **Eliminate all implication (\rightarrow) and rewrite**

1. $\forall x \neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
2. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
3. $\forall x \forall y \neg [\text{eats}(x, y) \wedge \neg \text{killed}(x)] \vee \text{food}(y)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
5. $\forall x \neg \text{eats}(\text{Anil}, x) \vee \text{eats}(\text{Harry}, x)$
6. $\forall x \neg [\neg \text{killed}(x)] \vee \text{alive}(x)$
7. $\forall x \neg \text{alive}(x) \vee \neg \text{killed}(x)$
8. $\text{likes}(\text{John}, \text{Peanuts})$.

Step-2: Conversion of FOL into CNF

Move negation (\neg) inwards and rewrite

1. $\forall x \neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
2. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
3. $\forall x \forall y \neg \text{eats}(x, y) \vee \text{killed}(x) \vee \text{food}(y)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
5. $\forall x \neg \text{eats}(\text{Anil}, x) \vee \text{eats}(\text{Harry}, x)$
6. $\forall x \text{killed}(x) \vee \text{alive}(x)$
7. $\forall x \neg \text{alive}(x) \vee \neg \text{killed}(x)$
8. $\text{likes}(\text{John}, \text{Peanuts}).$

- **Rename variables or standardize variables**

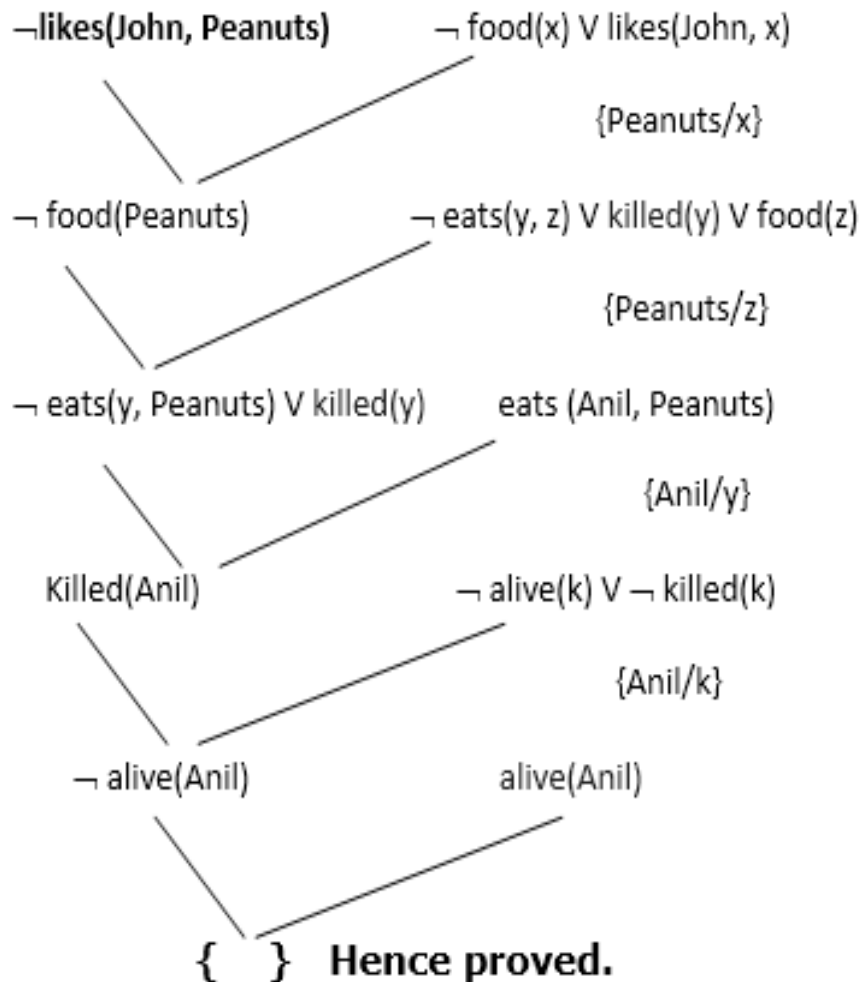
1. $\forall x \neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
2. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
3. $\forall y \forall z \neg \text{eats}(y, z) \vee \text{killed}(y) \vee \text{food}(z)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
5. $\forall w \neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$
6. $\forall g \neg \text{killed}(g) \vee \text{alive}(g)$
7. $\forall k \neg \text{alive}(k) \vee \neg \text{killed}(k)$
8. $\text{likes}(\text{John}, \text{Peanuts})$.

- **Eliminate existential instantiation quantifier by elimination**
- **Drop Universal quantifiers**
 1. $\neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
 2. $\text{food}(\text{Apple})$
 3. $\text{food}(\text{vegetables})$
 4. $\neg \text{eats}(y, z) \vee \text{killed}(y) \vee \text{food}(z)$
 5. $\text{eats}(\text{Anil}, \text{Peanuts})$
 6. $\text{alive}(\text{Anil})$
 7. $\neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$
 8. $\text{killed}(g) \vee \text{alive}(g)$
 9. $\neg \text{alive}(k) \vee \neg \text{killed}(k)$
 10. $\text{likes}(\text{John}, \text{Peanuts})$.

Step-3: Negate the statement to be proved

$\neg \text{likes}(\text{John}, \text{Peanuts})$

Step-4: Draw Resolution graph



1. $\neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
2. $\text{food}(\text{Apple})$
3. $\text{food}(\text{vegetables})$
4. $\neg \text{eats}(y, z) \vee \text{killed}(y) \vee \text{food}(z)$
5. $\text{eats}(\text{Anil}, \text{Peanuts})$
6. $\text{alive}(\text{Anil})$
7. $\neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$
8. $\text{killed}(g) \vee \text{alive}(g)$
9. $\neg \text{alive}(k) \vee \neg \text{killed}(k)$
10. $\text{likes}(\text{John}, \text{Peanuts})$.

Procedural and Declarative Knowledge

- We can express the knowledge in various forms to the inference engine in the computer system to solve the problems.
- There are two important representations of knowledge - Procedural knowledge and Declarative knowledge.
- The basic difference between procedural and declarative knowledge is that procedural knowledge gives the control information along with the knowledge, whereas declarative knowledge just provides the knowledge but not the control information to implement the knowledge.

Procedural Knowledge

- Procedural or imperative knowledge clarifies how to perform a certain task.
- It lays down the steps to perform.
- Thus, the procedural knowledge provides the essential control information required to implement the knowledge.
- We must have knowledge of
 - ✓ How to perform?
 - ✓ How to operate?
 - ✓ How to use procedures?
- Example – How to make coffee

Declarative Knowledge

- Also known as **Descriptive knowledge**
- Is the type of knowledge which tells the **basic knowledge about something**, and it is more popular than Procedural Knowledge
- It is **facts-based knowledge**
- **Static in nature**
- **Example –**
Sun rises in the east.

Examples

- **Employee ID : 112011**
- **Employee age : 28 Years**
- **Employee Tax**