

# ITDO6014

## AI AND DS-1

# Knowledge-Based Agent

2

- An intelligent agent needs knowledge about the real world for taking decisions and reasoning to act efficiently.
- Knowledge-based agents are those agents who have the capability of maintaining an internal state of knowledge, reason over that knowledge, update their knowledge after observations and take actions. These agents can represent the world with some formal representation and act intelligently.
- Knowledge-based agents are composed of two main parts:
  - Knowledge-base and
  - Inference system.

# Knowledge-Based Agent

3

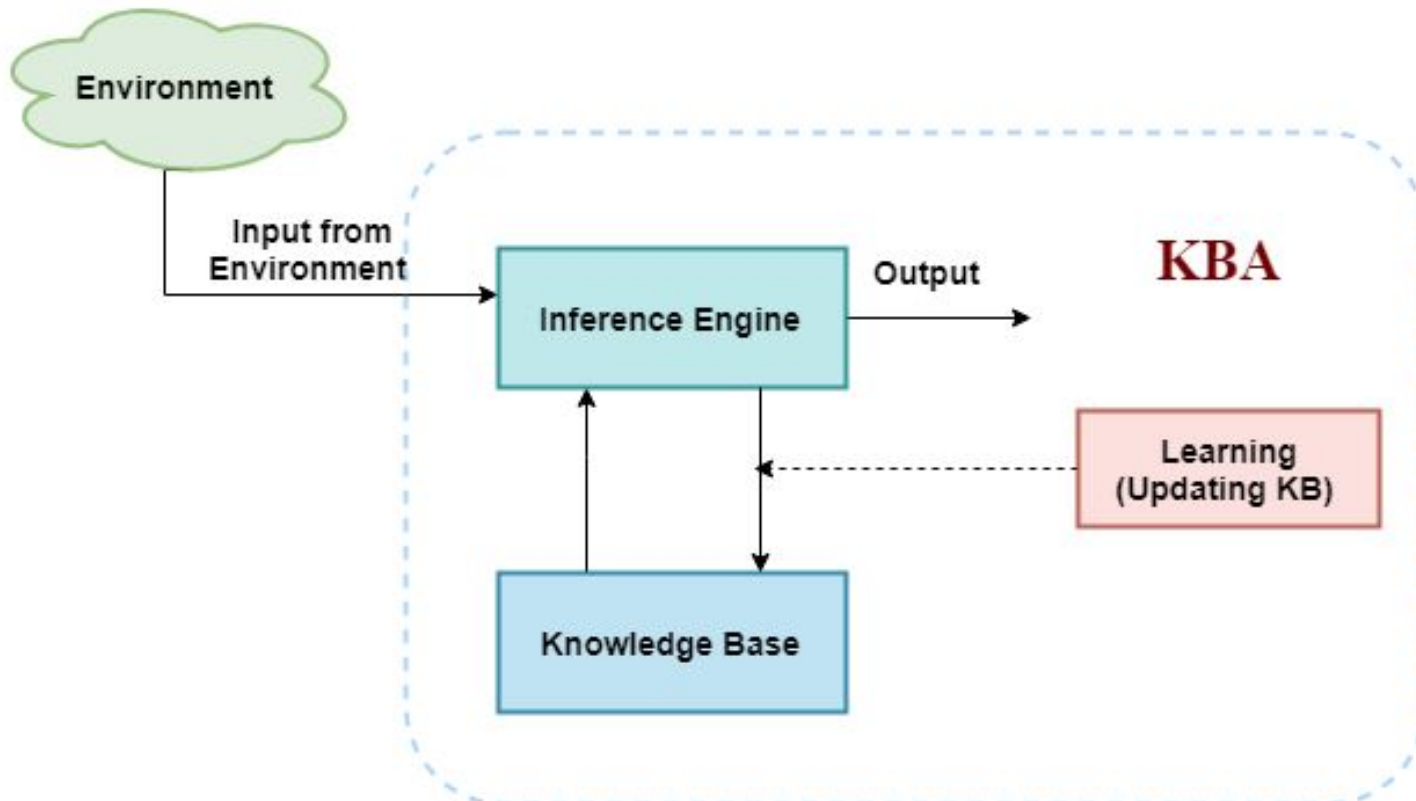
- A knowledge-based agent must be able to do the following:
- An agent should be able to represent states, actions, etc.
- An agent should be able to incorporate new percepts
- An agent can update the internal representation of the world
- An agent can deduce the internal representation of the world
- An agent can deduce appropriate actions

# Knowledge-Based Agent

4

- The architecture of knowledge-based agent:

□



# Knowledge-Based Agent

5

- The above diagram is representing a generalized architecture for a knowledge-based agent. The knowledge-based agent (KBA) take input from the environment by perceiving the environment. The input is taken by the inference engine of the agent and which also communicate with KB to decide as per the knowledge store in KB. The learning element of KBA regularly updates the KB by learning new knowledge.

# Knowledge-Based Agent

6

- Knowledge base: Knowledge-base is a central component of a knowledge-based agent, it is also known as KB. It is a collection of sentences (here 'sentence' is a technical term and it is not identical to sentence in English). These sentences are expressed in a language which is called a knowledge representation language. The Knowledge-base of KBA stores fact about the world.
- Knowledge-base is required for updating knowledge for an agent to learn with experiences and take action as per the knowledge.

# Knowledge-Based Agent

7

- Inference system: Inference means deriving new sentences from old. Inference system allows us to add a new sentence to the knowledge base. A sentence is a proposition about the world. Inference system applies logical rules to the KB to deduce new information.
- Inference system generates new facts so that an agent can update the KB. An inference system works mainly in two rules which are given as:
  - Forward chaining
  - Backward chaining

# Knowledge-Based Agent

8

- ❑ **Following are three operations which are performed by KBA in order to show the intelligent behavior:**
- ❑ **TELL:** This operation tells the knowledge base what it perceives from the environment.
- ❑ **ASK:** This operation asks the knowledge base what action it should perform.
- ❑ **Perform:** It performs the selected action.



# Knowledge-Based Agent

9

- Following is the structure outline of a generic knowledge-based agents program:
- function KB-AGENT(percept):
- persistent: 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

# Knowledge-Based Agent

10

- The knowledge-based agent takes percept as input and returns an action as output. The agent maintains the knowledge base, KB, and it initially has some background knowledge of the real world. It also has a counter to indicate the time for the whole process, and this counter is initialized with zero.
- Each time when the function is called, it performs its three operations:
  - Firstly it TELLS the KB what it perceives.
  - Secondly, it asks KB what action it should take
  - Third agent program TELLS the KB that which action was chosen.

# Knowledge-Based Agent

11

- The MAKE-PERCEPT-SENTENCE generates a sentence as setting that the agent perceived the given percept at the given time.
- The MAKE-ACTION-QUERY generates a sentence to ask which action should be done at the current time.
- MAKE-ACTION-SENTENCE generates a sentence which asserts that the chosen action was executed.

# Knowledge-Based Agent

12

- Various levels of knowledge-based agent:
- A knowledge-based agent can be viewed at different levels which are given below:
- 1. Knowledge level
- Knowledge level is the first level of knowledge-based agent, and in this level, we need to specify what the agent knows, and what the agent goals are. With these specifications, we can fix its behavior. For example, suppose an automated taxi agent needs to go from a station A to station B, and he knows the way from A to B, so this comes at the knowledge level.

# Knowledge-Based Agent

13

- 2. Logical level:
  - At this level, we understand that how the knowledge representation of knowledge is stored. At this level, sentences are encoded into different logics. At the logical level, an encoding of knowledge into logical sentences occurs. At the logical level we can expect to the automated taxi agent to reach to the destination B.
- 3. Implementation level:
  - This is the physical representation of logic and knowledge. At the implementation level agent perform actions as per logical and knowledge level. At this level, an automated taxi agent actually implement his knowledge and logic so that he can reach to the destination.

# Knowledge-Based Agent

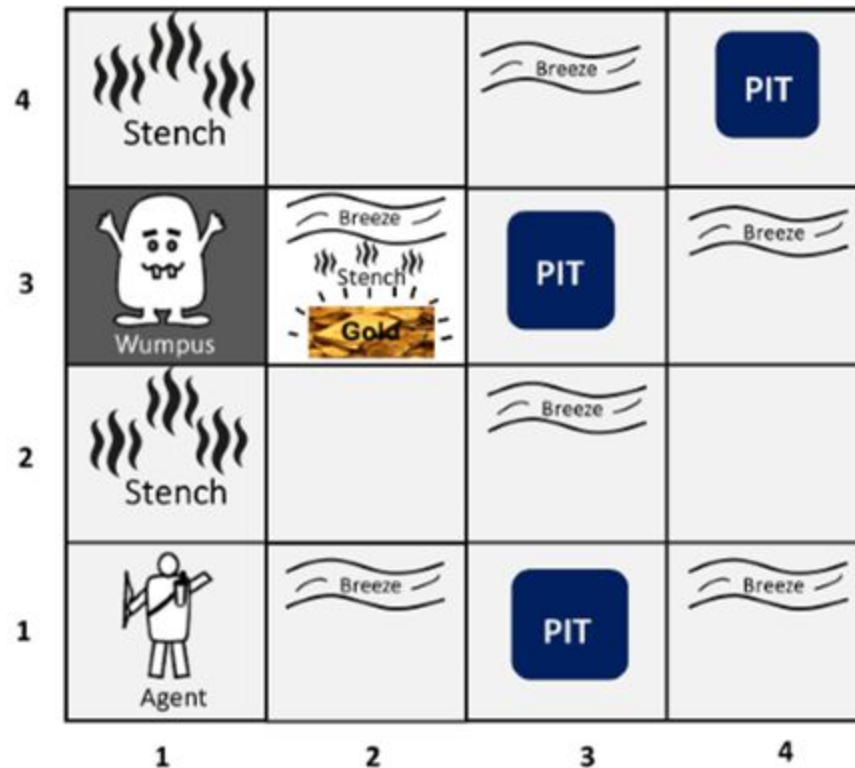
14

- Approaches to designing a knowledge-based agent:
- There are mainly two approaches to build a knowledge-based agent:
- **1. Declarative approach:** We can create a knowledge-based agent by initializing with an empty knowledge base and telling the agent all the sentences with which we want to start with. This approach is called Declarative approach.
- **2. Procedural approach:** In the procedural approach, we directly encode desired behavior as a program code. Which means we just need to write a program that already encodes the desired behavior or agent.

# Knowledge-Based Agent

15

- The Wumpus World in Artificial intelligence



# Knowledge-Based Agent

16

- **There are also some components which can help the agent to navigate the cave. These components are given as follows:**
- The rooms adjacent to the Wumpus room are smelly, so that it would have some stench.
- The room adjacent to PITs has a breeze, so if the agent reaches near to PIT, then he will perceive the breeze.
- There will be glitter in the room if and only if the room has gold.
- The Wumpus can be killed by the agent if the agent is facing to it, and Wumpus will emit a horrible scream which can be heard anywhere in the cave.



# Knowledge-Based Agent

17

- PEAS description of Wumpus world:
- To explain the Wumpus world we have given PEAS description as below:
- Performance measure:
- +1000 reward points if the agent comes out of the cave with the gold.
- -1000 points penalty for being eaten by the Wumpus or falling into the pit.
- -1 for each action, and -10 for using an arrow.
- The game ends if either agent dies or came out of the cave.

# Knowledge-Based Agent

18

- Actuators:
- Left turn,
- Right turn
- Move forward
- Grab
- Release
- Shoot.

# Knowledge-Based Agent

19

- Sensors: The agent will perceive the **stench** if he is in the room adjacent to the Wumpus. (Not diagonally).
- The agent will perceive **breeze** if he is in the room directly adjacent to the Pit.
- The agent will perceive the **glitter** in the room where the gold is present.
- The agent will perceive the **bump** if he walks into a wall.
- When the Wumpus is shot, it emits a horrible **scream** which can be perceived anywhere in the cave.
- These percepts can be represented as five element list, in which we will have different indicators for each sensor.
- Example if agent perceives stench, breeze, but no glitter, no bump, and no scream then it can be represented as:  
**[Stench, Breeze, None, None, None]**.

# Knowledge-Based Agent

20

- The Wumpus world Properties:
- **Partially observable:** The Wumpus world is partially observable because the agent can only perceive the close environment such as an adjacent room.
- **Deterministic:** It is deterministic, as the result and outcome of the world are already known.
- **Sequential:** The order is important, so it is sequential.
- **Static:** It is static as Wumpus and Pits are not moving.
- **Discrete:** The environment is discrete.
- **One agent:** The environment is a single agent as we have one agent only and Wumpus is not considered as an agent.

# Knowledge-Based Agent

21

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 ok	2,2	3,2	4,2
1,1 A ok	2,1 ok	3,1	4,1

(a)

Room is Safe, No  
Stench,  
No Breeze

A = Agent  
B = Agent  
G = Glitter, Gold  
ok = Safe, Square  
P = Pit  
S = Stench  
V = Visited  
W = Wumpus

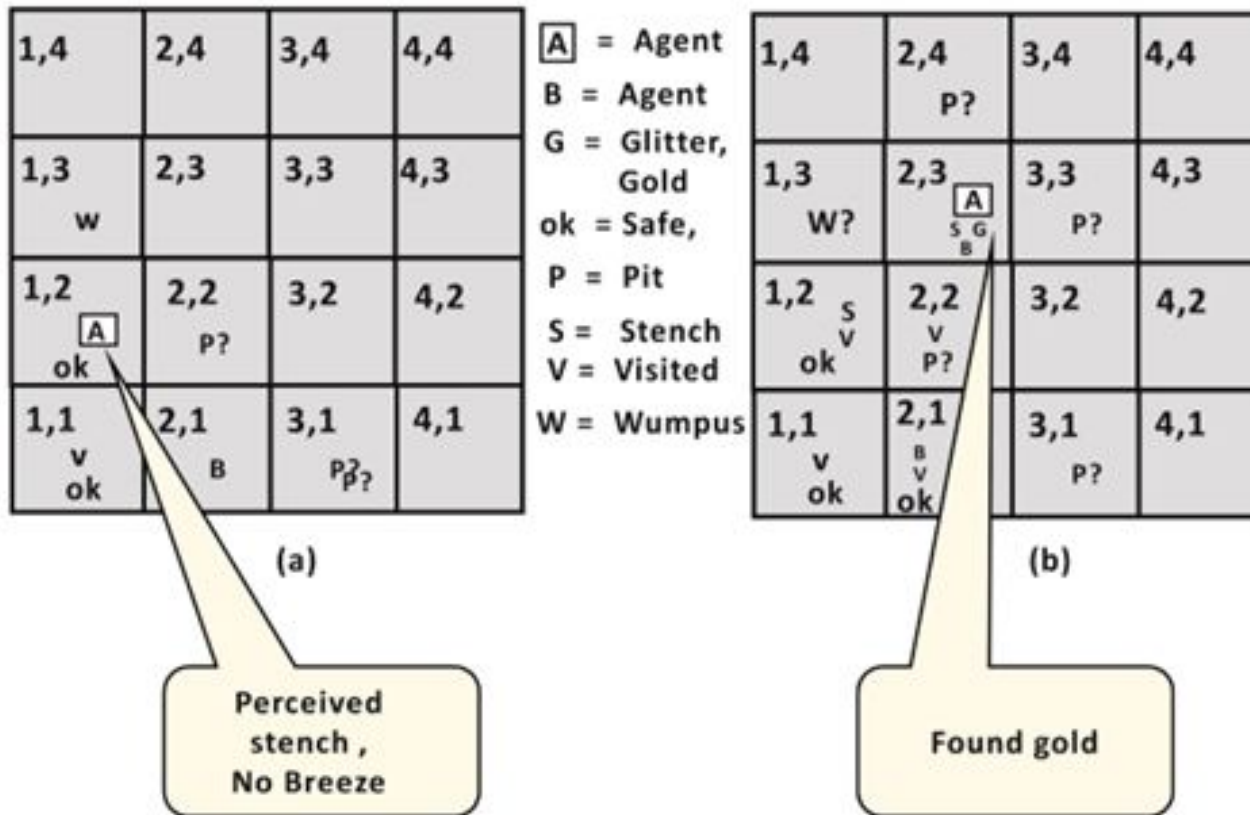
1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 ok	2,2 P?	3,2	4,2
1,1 v ok	2,1 A B ok	3,1 P?	4,1

(b)

Perceived Breeze,  
Adjacent room is not  
Safe Go Back

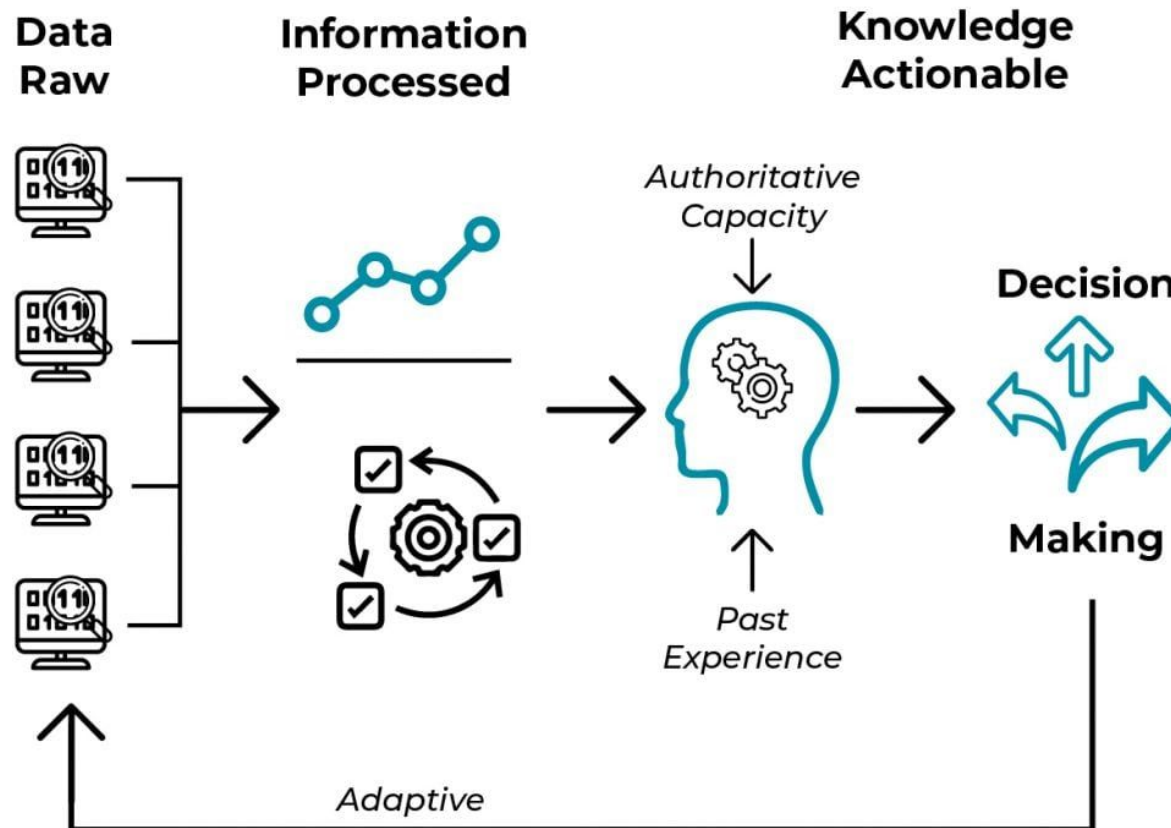
# Knowledge-Based Agent

22



# Knowledge Representation

23



# Knowledge Representation

24

- 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.** Hence we can describe Knowledge representation as following:
- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.



# Knowledge Representation

25

- 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 such as diagnosis a medical condition or communicating with humans in natural language.
- It is also a way which describes how we can represent knowledge in artificial intelligence. 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.

# Knowledge Representation

26

- 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 behavior which involves knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.

# Knowledge Representation

27

- **Facts:** Facts are the truths about the real world and what we represent.

# Knowledge Representation

28

- Approaches to knowledge representation:
- There are mainly four approaches to knowledge representation, which are given below:
  - 1. Simple relational knowledge:
  - It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.
  - This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.

# Knowledge Representation

29

- **Example: The following is the simple relational knowledge representation.**

Player	Weight	Age
Player1	65	23
Player2	58	18
Player3	75	24

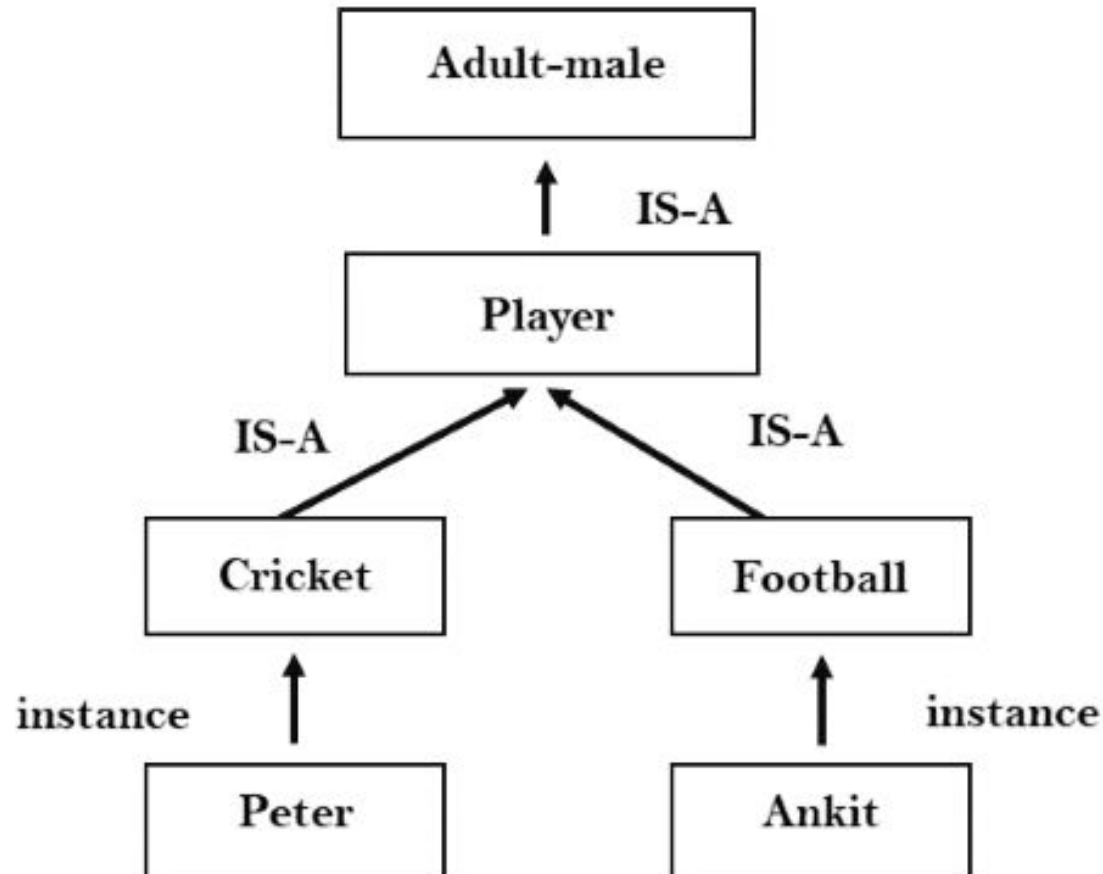
# Knowledge Representation

30

- 2. Inheritable knowledge:
- In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.
- All classes should be arranged in a generalized form or a hierarchical manner.
- In this approach, we apply inheritance property.
- Elements inherit values from other members of a class.

# Knowledge Representation

31



# Knowledge Representation

32

- Inferential knowledge:
- Inferential knowledge approach represents knowledge in the form of formal logics.
- This approach can be used to derive more facts.
- It guaranteed correctness.
- **Example:** Let's suppose there are two statements:
  - Marcus is a man
  - All men are mortalThen it can represent as;  
**man(Marcus)**  
 **$\forall x = \text{man}(x) \text{ -----} \rightarrow \text{mortal}(x)$**



# Knowledge Representation

33

- Techniques of knowledge representation
- There are mainly four ways of knowledge representation which are given as follows:
  - Logical Representation
  - Semantic Network Representation
  - Frame Representation

# Knowledge Representation

34

- 1. Logical Representation
- Logical representation is a language with some concrete rules which deals with propositions and has no ambiguity in representation. Logical representation means drawing a conclusion based on various conditions. This representation lays down some important communication rules. It consists of precisely defined syntax and semantics which supports the sound inference. Each sentence can be translated into logics using syntax and semantics.

# Knowledge Representation

35

- Syntax:
- Syntaxes are the rules which decide how we can construct legal sentences in the logic.
- It determines which symbol we can use in knowledge representation.
- How to write those symbols.

# Knowledge Representation

36

- Semantics:
- Semantics are the rules by which we can interpret the sentence in the logic.
- Semantic also involves assigning a meaning to each sentence.
- Logical representation can be categorised into mainly two logics:
  - Propositional Logics
  - Predicate logics

# Knowledge Representation

37

- 2. Semantic Network Representation
- Semantic networks are alternative of predicate logic for knowledge representation. In Semantic networks, we can 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. Semantic networks can categorize the object in different forms and can also link those objects. Semantic networks are easy to understand and can be easily extended.

# Knowledge Representation

38

- This representation consist of mainly two types of relations:
- IS-A relation (Inheritance)
- Kind-of-relation
- **Example:** Following are some statements which we need to represent in the form of nodes and arcs.

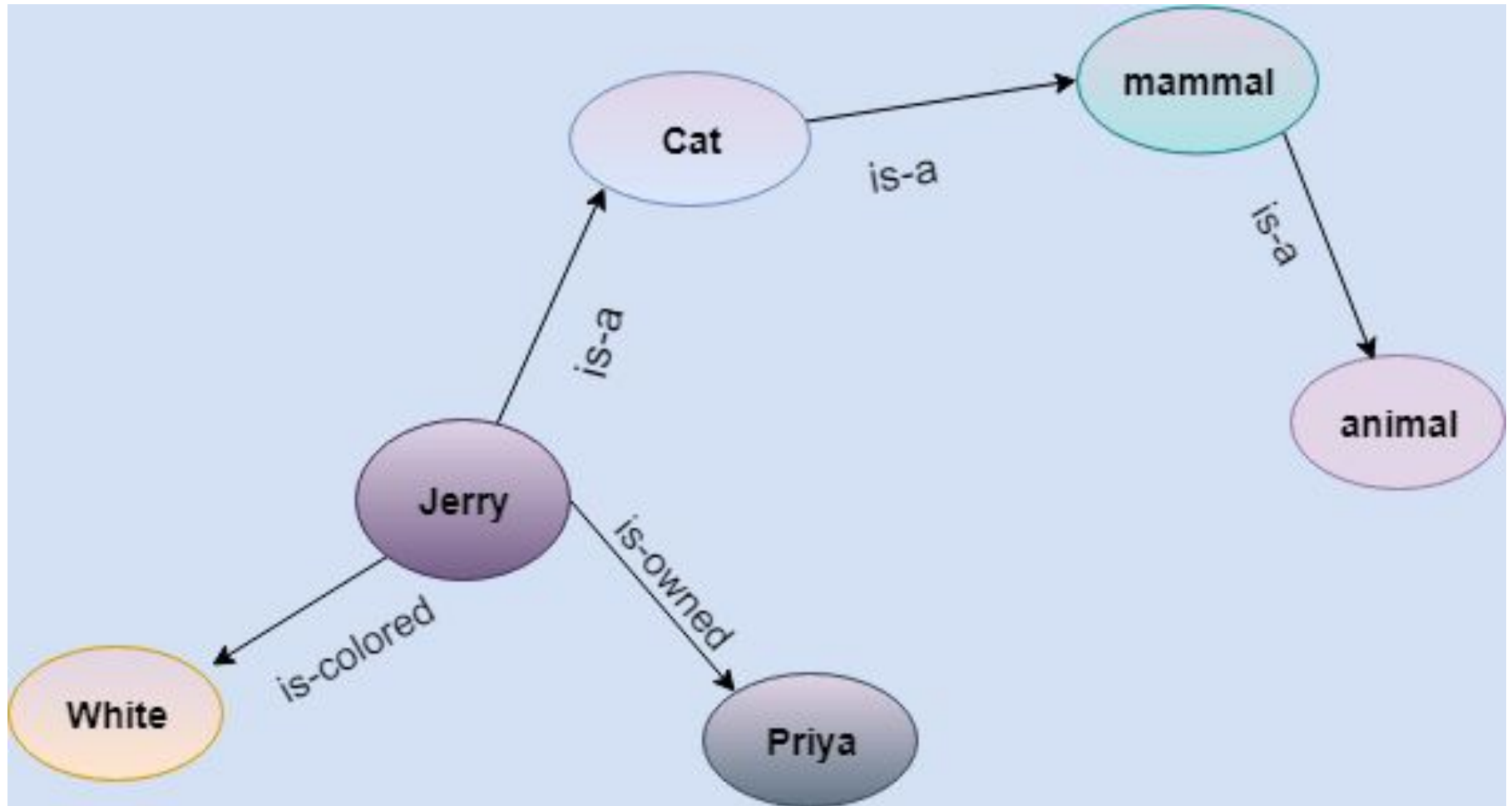
# Knowledge Representation

39

- Statements:
- Jerry is a cat.
- Jerry is a mammal
- Jerry is owned by Priya.
- Jerry is brown colored.
- All Mammals are animal.

# Knowledge Representation

40





# Knowledge Representation

41

- 3. Frame Representation
- A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world. Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations. It consists of a collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facets.

# Knowledge Representation

42

- Example: 1
- Let's take an example of a frame for a book

Slots	Filters
<b>Title</b>	Artificial Intelligence
<b>Genre</b>	Computer Science
<b>Author</b>	Peter Norvig
<b>Edition</b>	Third Edition
<b>Year</b>	1996
<b>Page</b>	1152

# Knowledge Representation

43

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

# Knowledge Representation

44

- **Following are some basic facts about propositional logic:**
- Propositional logic is also called Boolean logic as it works on 0 and 1.
- In propositional logic, we 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 consists of an object, relations or function, and **logical connectives**.

# Knowledge Representation

45

- These connectives are also called logical operators.
- The propositions and connectives are the basic elements of the propositional logic.
- Connectives can be said as a logical operator which connects two sentences.
- A proposition formula which is always true is called **tautology**, and it is also called a valid sentence.
- A proposition formula which is always false is called **Contradiction**.

# Knowledge Representation

46

- A proposition formula which has both true and false values is called
- 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:
  - **Atomic Propositions**
  - **Compound propositions**

# Knowledge Representation

47

- **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.
- **Example:**
  - a)  $2+2$  is 4, it is an atomic proposition as it is a **true** fact.
  - b) "The Sun is cold" is also a proposition as it is a **false** fact.

# Knowledge Representation

48

- **Compound proposition:** Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.
- **Example:**
  - a) "It is raining today, and street is wet."
  - b) "Ankit is a doctor, and his clinic is in Mumbai."



# Knowledge Representation

49

- **Conjunction:** A sentence which has  $\wedge$  connective such as,  $P \wedge Q$  is called a conjunction.

**Example:** Rohan is intelligent and hardworking. It can be written as,

**P = Rohan is intelligent,**

**Q = Rohan is hardworking.  $\rightarrow P \wedge Q$ .**

- **Disjunction:** A sentence which has  $\vee$  connective, such as  $P \vee Q$ . is called disjunction, where P and Q are the propositions.

**Example: "Ritika is a doctor or Engineer",**

Here  $P = \text{Ritika is Doctor}$ .  $Q = \text{Ritika is Doctor}$ , so we can write it as  $P \vee Q$ .

# Knowledge Representation

50

- **Implication:** A sentence such as  $P \rightarrow Q$ , is called an implication. Implications are also known as if-then rules. It can be represented as

**If** it is raining, then the street is wet.

Let  $P =$  It is raining, and  $Q =$  Street is wet, so it is represented as  $P \rightarrow Q$

- **Biconditional:** A sentence such as  $P \Leftrightarrow Q$  is a **Biconditional sentence, example If I am breathing, then I am alive**

$P =$  I am breathing,  $Q =$  I am alive, it can be represented as  $P \Leftrightarrow Q$ .

# Knowledge Representation

51

- Following is the summarized table for Propositional Logic Connectives:

Connective symbols	Word	Technical term	Example
$\wedge$	AND	Conjunction	$A \wedge B$
$\vee$	OR	Disjunction	$A \vee B$
$\rightarrow$	Implies	Implication	$A \rightarrow B$
$\Leftrightarrow$	If and only if	Biconditional	$A \Leftrightarrow B$
$\neg$ or $\sim$	Not	Negation	$\neg A$ or $\neg B$

# Knowledge Representation

52

- Truth Table:
- In propositional logic, we need to know the truth values of propositions in all possible scenarios. We can combine all the possible combination with logical connectives, and the representation of these combinations in a tabular format is called **Truth table**. Following are the truth table for all logical connectives:

# Knowledge Representation

53

**For Negation:**

P	$\neg P$
True	False
False	True

**For Conjunction:**

P	Q	$P \wedge Q$
True	True	True
True	False	False
False	True	False
False	False	False

# Knowledge Representation

54

**For disjunction:**

P	Q	$P \vee Q$
True	True	True
False	True	True
True	False	True
False	False	False

**For Implication:**

P	Q	$P \rightarrow Q$
True	True	True
True	False	False
False	True	True
False	False	True

**For Biconditional:**

P	Q	$P \leftrightarrow Q$
True	True	True
True	False	False
False	True	False
False	False	True

# Knowledge Representation

55

- Truth table with three propositions:
- We can build a proposition composing three propositions P, Q, and R. This truth table is made-up of 8n Tuples as we have taken three proposition symbols.

P	Q	R	$\neg R$	$P \vee Q$	$P \vee Q \rightarrow \neg R$
True	True	True	False	True	False
True	True	False	True	True	True
True	False	True	False	True	False
True	False	False	True	True	True
False	True	True	False	True	False
False	True	False	True	True	True
False	False	True	False	False	True
False	False	False	True	False	True

# Knowledge Representation

56

- Logical equivalence:
- Logical equivalence is one of the features of propositional logic. Two propositions are said to be logically equivalent if and only if the columns in the truth table are identical to each other.
- Let's take two propositions A and B, so for logical equivalence, we can write it as  $A \Leftrightarrow B$ . In below truth table we can see that column for  $\neg A \vee B$  and  $A \rightarrow B$ , are identical hence A is Equivalent to B



# Knowledge Representation

57

A	B	$\neg A$	$\neg A \vee B$	$A \rightarrow B$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

# Knowledge Representation

58

- Properties of Operators:
- **Commutativity:**
  - $P \wedge Q = Q \wedge P$ , or
  - $P \vee Q = Q \vee P$ .
- **Associativity:**
  - $(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$ ,
  - $(P \vee Q) \vee R = P \vee (Q \vee R)$
- **Double-negation elimination:**
  - $\neg(\neg P) = P$ .

# Knowledge Representation

59

## □ Identity element:

- $P \wedge \text{True} = P,$
- $P \vee \text{True} = \text{True}.$

## □ 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:

- $\neg (P \wedge Q) = (\neg P) \vee (\neg Q)$
- $\neg (P \vee Q) = (\neg P) \wedge (\neg Q).$

# Knowledge Representation

60

- Limitations of Propositional logic:
- We cannot represent relations like ALL, some, or none with propositional logic. Example:
  - **All the girls are intelligent.**
  - **Some apples are sweet.**

# Rules of Inference in Artificial intelligence

61

- Inference:
- In artificial intelligence, we need intelligent computers which can create new logic from old logic or by evidence, **so generating the conclusions from evidence and facts is termed as Inference.**

# Knowledge Representation

62

- Inference rules: In inference rules, the implication among all the connectives plays an important role. Following are some terminologies related to inference rules:
- **Implication:** It is one of the logical connectives which can be represented as  $P \rightarrow Q$ . It is a Boolean expression.
- **Converse:** The converse of implication, which means the right-hand side proposition goes to the left-hand side and vice-versa. It can be written as  $Q \rightarrow P$ .
- **Contrapositive:** The negation of converse is termed as contrapositive, and it can be represented as  $\neg Q \rightarrow \neg P$ .
- **Inverse:** The negation of implication is called inverse. It can be represented as  $\neg P \rightarrow \neg Q$ .

# Knowledge Representation

63

P	Q	$P \rightarrow Q$	$Q \rightarrow P$	$\neg Q \rightarrow \neg P$	$\neg P \rightarrow \neg Q$
T	T	T	T	T	T
T	F	F	T	F	T
F	T	T	F	T	F
F	F	T	T	T	T

# Knowledge Representation

64

- 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.



# Knowledge Representation

65

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



# Knowledge Representation

66

- 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: 
$$\frac{P \rightarrow Q, \sim Q}{\sim P}$$

- **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**"  $\Rightarrow \sim P$

# Knowledge Representation

67

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

# Knowledge Representation

68

- 3. Hypothetical Syllogism:
- The Hypothetical Syllogism rule state that if  $P \rightarrow R$  is true whenever  $P \rightarrow Q$  is true, and  $Q \rightarrow R$  is true. It can be represented as the following notation:
- **Example:**
- **Statement-1:** If you have my home key then you can unlock my home.  $P \rightarrow Q$
- **Statement-2:** If you can unlock my home then you can take my money.  $Q \rightarrow R$
- **Conclusion:** If you have my home key then you can take my money.  $P \rightarrow R$

# Knowledge Representation

69

P	Q	R	$P \rightarrow Q$	$Q \rightarrow R$	$P \rightarrow R$	
0	0	0	1	1	1	←
0	0	1	1	1	1	←
0	1	0	1	0	1	
0	1	1	1	1	1	←
1	0	0	0	1	1	
1	0	1	0	1	1	
1	1	0	1	0	0	
1	1	1	1	1	1	←

# Knowledge Representation

70

- 4. Disjunctive Syllogism:
- The Disjunctive syllogism rule state that if  $P \vee Q$  is true, and  $\neg P$  is true, then  $Q$  will be true. It can be represented as:

Notation of Disjunctive syllogism:  $\frac{P \vee Q, \neg P}{Q}$

**Example:**

**Statement-1:** Today is Sunday or Monday.  $\implies P \vee Q$

**Statement-2:** Today is not Sunday.  $\implies \neg P$

**Conclusion:** Today is Monday.  $\implies Q$

**Proof by truth-table:**

P	Q	$\neg P$	$P \vee Q$
0	0	1	0
0	1	1	1
1	0	0	1
1	1	0	1

# Knowledge Representation

71

## 5. Addition:

The Addition rule is one the common inference rule, and it states that If P is true, then  $P \vee Q$  will be true.

Notation of Addition:  $\frac{P}{P \vee Q}$

### Example:

**Statement:** I have a vanilla ice-cream.  $\Rightarrow P$

**Statement-2:** I have Chocolate ice-cream.

**Conclusion:** I have vanilla or chocolate ice-cream.  $\Rightarrow (P \vee Q)$

### Proof by Truth-Table:

P	Q	$P \vee Q$
0	0	0
1	0	1
0	1	1
1	1	1

# Knowledge Representation

72


## 6. Simplification:

The simplification rule state that if  $P \wedge Q$  is true, then  $Q$  or  $P$  will also be true. It can be represented as:

Notation of Simplification rule:  $\frac{P \wedge Q}{Q}$  Or  $\frac{P \wedge Q}{P}$

### Proof by Truth-Table:

P	Q	$P \wedge Q$
0	0	0
1	0	0
0	1	0
1	1	1





# First-Order Logic in Artificial intelligence

73

- In the topic of Propositional logic, we have seen that how to represent statements using propositional logic. But unfortunately, in propositional logic, we can only represent the facts, which are either true or false. PL is not sufficient to represent the complex sentences or natural language statements. The propositional logic has very limited expressive power. Consider the following sentence, which we cannot represent using PL logic.
- **"Some humans are intelligent", or**
- **"Sachin likes cricket."**
- To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as first-order logic.

# First-Order Logic in Artificial intelligence

74

- First-Order logic
- First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.
- FOL is sufficiently expressive to represent the natural language statements in a concise way.
- First-order logic is also known as **Predicate logic or First-order predicate logic**. First-order logic is a powerful language that develops information about the objects in a more easy way and can also express the relationship between those objects.

# First-Order Logic in Artificial intelligence

75

- First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:
  - **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, .....
  - **Relations:** It can be unary relation such as: red, round, is adjacent, or n-any relation such as: the sister of, brother of, has color, comes between
  - **Function:** Father of, best friend, third inning of, end of, .....
- As a natural language, first-order logic also has two main parts:
  - **Syntax**
  - **Semantics**

# First-Order Logic in Artificial intelligence

76

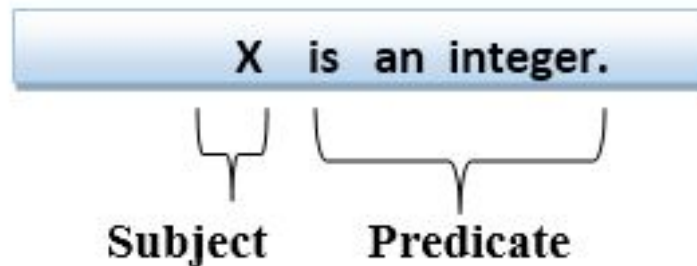
- Basic Elements of First-order logic:
- Following are the basic elements of FOL syntax:

<b>Constant</b>	1, 2, A, John, Mumbai, cat,....
<b>Variables</b>	x, y, z, a, b,....
<b>Predicates</b>	Brother, Father, >,....
<b>Function</b>	sqrt, LeftLegOf, ....
<b>Connectives</b>	$\wedge$ , $\vee$ , $\neg$ , $\Rightarrow$ , $\Leftrightarrow$
<b>Equality</b>	$=$
<b>Quantifier</b>	$\forall$ , $\exists$

# First-Order Logic in Artificial intelligence

77

- **First-order logic statements can be divided into two parts:**
- **Subject:** Subject is the main part of the statement.
- **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.
- **Consider the statement: "x is an integer."**, it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



# First-Order Logic in Artificial intelligence

78

- Atomic sentences:
- Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as **Predicate (term1, term2, ....., term n)**.
- **Example: Ravi and Ajay are brothers:  $\Rightarrow$  Brothers(Ravi, Ajay).**  
**Simba is a Lion:  $\Rightarrow$  Lion (Simba).**
- Complex Sentences:
- Complex sentences are made by combining atomic sentences using connectives.

# First-Order Logic in Artificial intelligence

79

- Quantifiers in First-order logic:
- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:
  - **Universal Quantifier, (for all, everyone, everything)**
  - **Existential quantifier, (for some, at least one).**

# First-Order Logic in Artificial intelligence

80

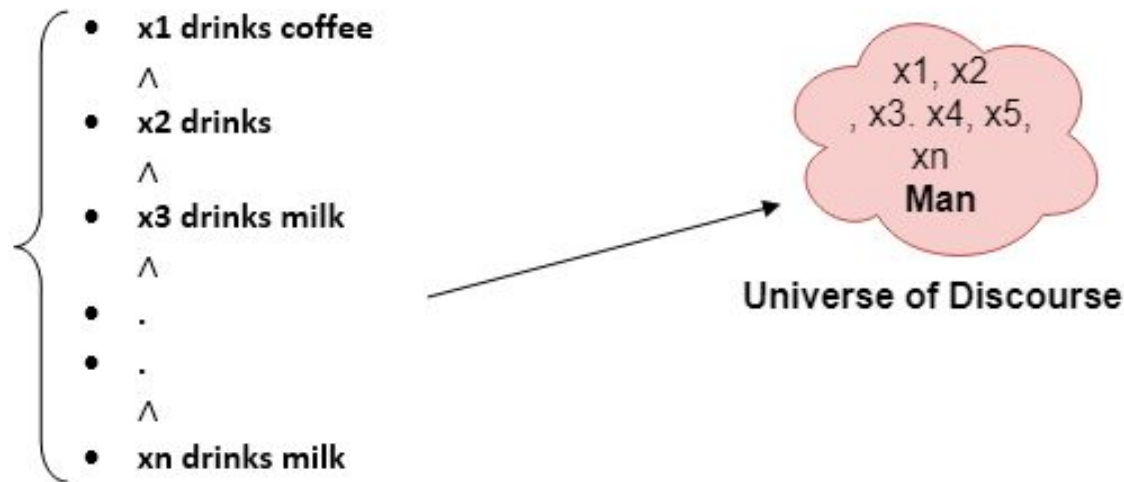
- Universal Quantifier:
- Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.
- The Universal quantifier is represented by a symbol  $\forall$ , which resembles an inverted A.
- Note: In universal quantifier we use implication " $\rightarrow$ ".
- If  $x$  is a variable, then  $\forall x$  is read as:
  - **For all  $x$**
  - **For each  $x$**
  - **For every  $x$ .**



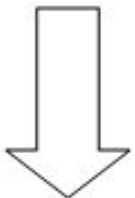
# First-Order Logic in Artificial intelligence

81

- Example:
- **All man drink coffee.**
- Let a variable  $x$  which refers to a man so all  $x$  can be represented in UOD as below:



$\forall x \text{ man}(x) \rightarrow \text{drink}(x, \text{coffee}).$   
It will be read as: There are all  $x$  where  $x$  is a man who drink coffee.



So in shorthand notation, we can write it as :

# First-Order Logic in Artificial intelligence

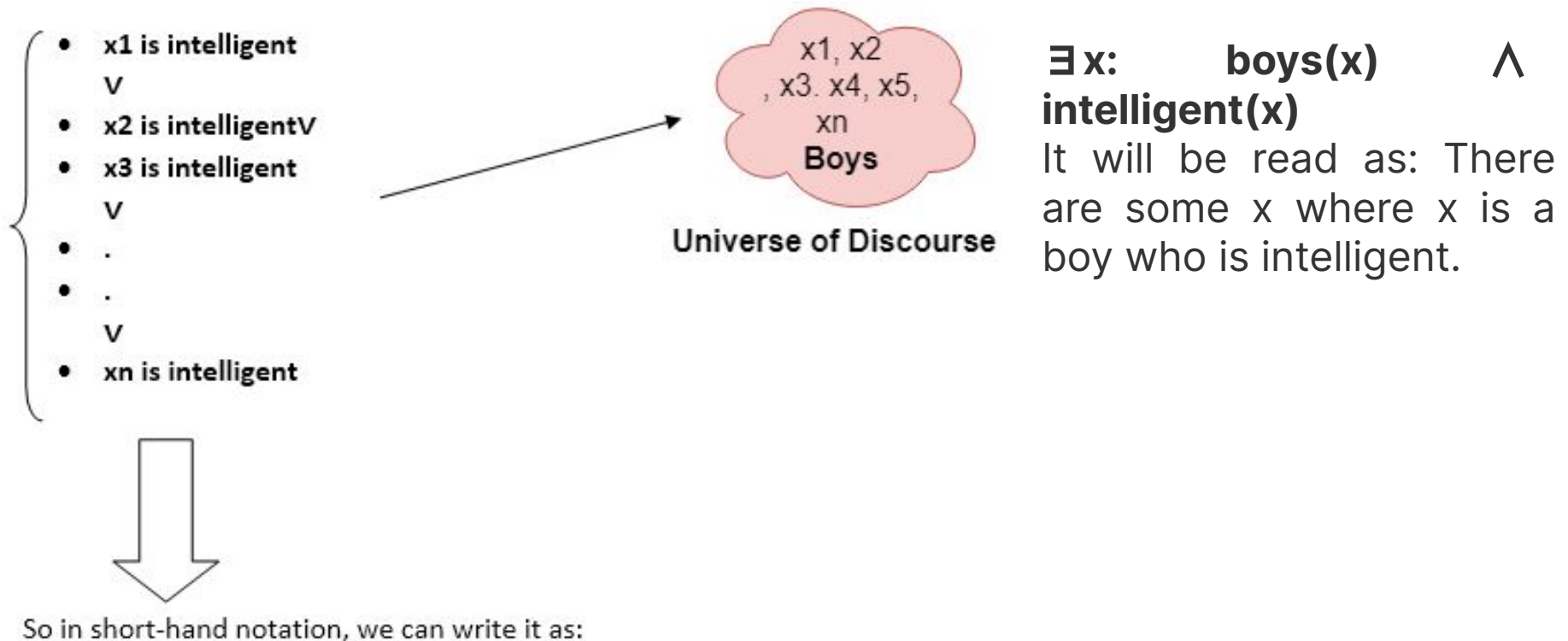
82

- Existential Quantifier:
- Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.
- It is denoted by the logical operator  $\exists$ , which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.
- Note: In Existential quantifier we always use AND or Conjunction symbol ( $\wedge$ ).
- If  $x$  is a variable, then existential quantifier will be  $\exists x$  or  $\exists (x)$ . And it will be read as:
- **There exists a 'x.'**
- **For some 'x.'**
- **For at least one 'x.'**

# First-Order Logic in Artificial intelligence

83

- Example:
- **Some boys are intelligent.**



# First-Order Logic in Artificial intelligence

84

- Points to remember:
- The main connective for universal quantifier  $\forall$  is implication  $\rightarrow$ .
- The main connective for existential quantifier  $\exists$  is and  $\wedge$ .
- Properties of Quantifiers:
- In universal quantifier,  $\forall x \forall y$  is similar to  $\forall y \forall x$ .
- In Existential quantifier,  $\exists x \exists y$  is similar to  $\exists y \exists x$ .
- $\exists x \forall y$  is not similar to  $\forall y \exists x$ .

# First-Order Logic in Artificial intelligence

85

- Some Examples of FOL using quantifier:

- **1. All birds fly.**

In this question the predicate is "**fly(bird).**"

And since there are all birds who fly so it will be represented as follows.

$$\forall x \text{ bird}(x) \rightarrow \text{fly}(x).$$

- **2. Every man respects his parent.**

In this question, the predicate is "**respect(x, y),**" where **x=man,** and **y= parent.**

Since there is every man so will use  $\forall$ , and it will be represented as follows:

$$\forall x \text{ man}(x) \rightarrow \text{respects}(x, \text{parent}).$$

# First-Order Logic in Artificial intelligence

86

## □ 3. Some boys play cricket.

In this question, the predicate is "**play(x, y)**," where **x= boys**, and **y= game**. Since there are some boys so we will use  **$\exists$** , and it will be represented as:

$$\exists x \text{ boys}(x) \wedge \text{play}(x, \text{cricket}).$$

## □ 4. Not all students like both Mathematics and Science.

In this question, the predicate is "**like(x, y)**," where **x= student**, and **y= subject**.

Since there are not all students, so we will use  **$\forall$  with negation**, so following representation for this:

$$\neg \forall (x) [ \text{student}(x) \rightarrow \text{like}(x, \text{Mathematics}) \wedge \text{like}(x, \text{Science}) ].$$

# Unification

87

- What is Unification?
- Unification is a process of making two different logical atomic expressions identical by finding a substitution. Unification depends on the substitution process.
- It takes two literals as input and makes them identical using substitution.
- Let  $\Psi_1$  and  $\Psi_2$  be two atomic sentences and  $\sigma$  be a unifier such that,  $\Psi_1\sigma = \Psi_2\sigma$ , then it can be expressed as **UNIFY**( $\Psi_1, \Psi_2$ ).

# Unification

88

- **Example: Find the MGU for  $\text{Unify}\{\text{King}(x), \text{King}(\text{John})\}$**
- Let  $\Psi_1 = \text{King}(x)$ ,  $\Psi_2 = \text{King}(\text{John})$ ,
- **Substitution  $\theta = \{\text{John}/x\}$**  is a unifier for these atoms and applying this substitution, and both expressions will be identical.
- The UNIFY algorithm is used for unification, which takes two atomic sentences and returns a unifier for those sentences (If any exist).
- Unification is a key component of all first-order inference algorithms.
- It returns fail if the expressions do not match with each other.
- The substitution variables are called Most General Unifier or MGU.



# Unification

89

- **E.g.** Let's say there are two different expressions,  $P(x, y)$ , and  $P(a, f(z))$ .
- In this example, we need to make both above statements identical to each other. For this, we will perform the substitution.
- $P(x, y)$ ..... (i)  
     $P(a, f(z))$ ..... (ii)
- Substitute  $x$  with  $a$ , and  $y$  with  $f(z)$  in the first expression, and it will be represented as  $a/x$  and  $f(z)/y$ .
- With both the substitutions, the first expression will be identical to the second expression and the substitution set will be:  $[a/x, f(z)/y]$ .

# Unification

90

- **Following are some basic conditions for unification:**
- Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified.
- Number of Arguments in both expressions must be identical.
- Unification will fail if there are two similar variables present in the same expression.

# Unification

91

- Implementation of the Algorithm
- **Step.1:** Initialize the substitution set to be empty.
- **Step.2:** Recursively unify atomic sentences:
- Check for Identical expression match.
- If one expression is a variable  $v_i$ , and the other is a term  $t_i$  which does not contain variable  $v_i$ , then:
  - Substitute  $t_i / v_i$  in the existing substitutions
  - Add  $t_i / v_i$  to the substitution setlist.
  - If both the expressions are functions, then function name must be similar, and the number of arguments must be the same in both the expression.

# Unification

92

- or each pair of the following atomic sentences find the most general unifier (If exist).
- 1. Find the MGU of  $\{p(f(a), g(Y)) \text{ and } p(X, X)\}$
- Sol:  $S_0 \Rightarrow$  Here,  $\Psi_1 = p(f(a), g(Y))$ , and  $\Psi_2 = p(X, X)$   
SUBST  $\theta = \{f(a) / X\}$   
 $S1 \Rightarrow \Psi_1 = p(f(a), g(Y))$ , and  $\Psi_2 = p(f(a), f(a))$   
SUBST  $\theta = \{f(a) / g(Y)\}$ , **Unification failed.**

# Unification

93

- **2. Find the MGU of  $\{p(b, X, f(g(Z)))$  and  $p(Z, f(Y), f(Y))\}$**
- Here,  $\Psi_1 = p(b, X, f(g(Z)))$ , and  $\Psi_2 = p(Z, f(Y), f(Y))$   
 $S_0 \Rightarrow \{p(b, X, f(g(Z))); p(Z, f(Y), f(Y))\}$   
 $\text{SUBST } \theta = \{b/Z\}$
- $S_1 \Rightarrow \{p(b, X, f(g(b))); p(b, f(Y), f(Y))\}$   
 $\text{SUBST } \theta = \{f(Y)/X\}$
- $S_2 \Rightarrow \{p(b, f(Y), f(g(b))); p(b, f(Y), f(Y))\}$   
 $\text{SUBST } \theta = \{g(b)/Y\}$
- $S_2 \Rightarrow \{p(b, f(g(b)), f(g(b))); p(b, f(g(b)), f(g(b)))\}$  **Unified Successfully.**  
**And Unifier =  $\{b/Z, f(Y)/X, g(b)/Y\}$ .**

# Unification

94

- **3. Find the MGU of  $\{p(X, X), \text{ and } p(Z, f(Z))\}$**
- Here,  $\Psi_1 = \{p(X, X), \text{ and } \Psi_2 = p(Z, f(Z))\}$   
 $S_0 \Rightarrow \{p(X, X), p(Z, f(Z))\}$   
 $\text{SUBST } \theta = \{X/Z\}$   
 $S1 \Rightarrow \{p(Z, Z), p(Z, f(Z))\}$   
 $\text{SUBST } \theta = \{f(Z) / Z\}, \text{ **Unification Failed.**}$

# Unification

95

- **4. Find the MGU of UNIFY(prime (1 1), prime(y))**
- Here,  $\Psi_1 = \{\text{prime}(1\ 1)\}$ , and  $\Psi_2 = \{\text{prime}(y)\}$   
 $S_0 \Rightarrow \{\text{prime}(1\ 1), \text{prime}(y)\}$   
SUBST  $\theta = \{1\ 1/y\}$
- $S_1 \Rightarrow \{\text{prime}(1\ 1), \text{prime}(1\ 1)\}$ , **Successfully unified.**  
**Unifier:  $\{1\ 1/y\}$ .**

# Unification

96

- **5. Find the MGU of  $Q(a, g(x, a), f(y)), Q(a, g(f(b), a), x)$**
- Here,  $\Psi_1 = Q(a, g(x, a), f(y))$ , and  $\Psi_2 = Q(a, g(f(b), a), x)$   
 $S_0 \Rightarrow \{Q(a, g(x, a), f(y)); Q(a, g(f(b), a), x)\}$   
SUBST  $\theta = \{f(b)/x\}$   
 $S_1 \Rightarrow \{Q(a, g(f(b), a), f(y)); Q(a, g(f(b), a), f(b))\}$
- SUBST  $\theta = \{b/y\}$   
 $S_1 \Rightarrow \{Q(a, g(f(b), a), f(b)); Q(a, g(f(b), a), f(b))\}$ , **Successfully Unified.**
- **Unifier:  $[f(b)/x, b/y]$ .**



# Unification

97

- **6. UNIFY(knows(Richard, x), knows(Richard, John))**
- Here,  $\Psi_1 = \text{knows(Richard, x)}$ , and  $\Psi_2 = \text{knows(Richard, John)}$   
 $S_0 \Rightarrow \{ \text{knows(Richard, x); knows(Richard, John)} \}$   
SUBST  $\theta = \{ \text{John/x} \}$   
 $S_1 \Rightarrow \{ \text{knows(Richard, John); knows(Richard, John)} \}$ , **Successfully Unified.**  
**Unifier:  $\{ \text{John/x} \}$ .**

# Resolution in FOL

98

- Resolution
- Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. It was invented by a Mathematician John Alan Robinson in the year 1965.
- Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**.
- **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**.

# Resolution in FOL

99

- Steps for Resolution:
- Conversion of facts into first-order logic.
- Convert FOL statements into CNF
- Negate the statement which needs to prove (proof by contradiction)
- Draw resolution graph (unification).

# Resolution in FOL

100

- Example:
  - **John likes all kind of food.**
  - **Apple and vegetable are food**
  - **Anything anyone eats and not killed is food.**
  - **Anil eats peanuts and still alive**
  - **Harry eats everything that Anil eats.**
  -
- Prove by resolution that John likes peanuts.**

# Resolution in FOL

101

## □ Step-1: Conversion of Facts into FOL

- In the first step we will convert all the given statements into its first order logic.

- 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.

# Resolution in FOL

102

- **Step-2: Conversion of FOL into CNF**
- In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.
- **Eliminate all implication ( $\rightarrow$ ) and rewrite ( $P \rightarrow Q = \sim P \vee Q$ )**
  - $\forall x \neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
  - $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
  - $\forall x \forall y \neg [\text{eats}(x, y) \wedge \neg \text{killed}(x)] \vee \text{food}(y)$
  - $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
  - $\forall x \neg \text{eats}(\text{Anil}, x) \vee \text{eats}(\text{Harry}, x)$
  - $\forall x \neg [\neg \text{killed}(x)] \vee \text{alive}(x)$
  - $\forall x \neg \text{alive}(x) \vee \neg \text{killed}(x)$
  - $\text{likes}(\text{John}, \text{Peanuts}).$

# Resolution in FOL

103

- **Move negation ( $\neg$ ) inwards and rewrite**
  - $\forall x \neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
  - $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
  - $\forall x \forall y \neg \text{eats}(x, y) \vee \text{killed}(x) \vee \text{food}(y)$
  - $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
  - $\forall x \neg \text{eats}(\text{Anil}, x) \vee \text{eats}(\text{Harry}, x)$
  - $\forall x \neg \text{killed}(x) \vee \text{alive}(x)$
  - $\forall x \neg \text{alive}(x) \vee \neg \text{killed}(x)$
  - $\text{likes}(\text{John}, \text{Peanuts}).$

# Resolution in FOL

104

## □ Rename variables or standardize variables

- $\forall x \neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
- $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
- $\forall y \forall z \neg \text{eats}(y, z) \vee \text{killed}(y) \vee \text{food}(z)$
- $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
- $\forall w \neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$
- $\forall g \neg \text{killed}(g) \vee \text{alive}(g)$
- $\forall k \neg \text{alive}(k) \vee \neg \text{killed}(k)$
- $\text{likes}(\text{John}, \text{Peanuts}).$



# Resolution in FOL

105

- **Eliminate existential instantiation quantifier by elimination.**

In this step, we will eliminate existential quantifier  $\exists$ , and this process is known as **Skolemization**. But in this example problem since there is no existential quantifier so all the statements will remain same in this step.

# Resolution in FOL

106

## □ **Drop Universal quantifiers.**

In this step we will drop all universal quantifier since all the statements are not implicitly quantified so we don't need it.

- $\neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
- $\text{food}(\text{Apple})$
- $\text{food}(\text{vegetables})$
- $\neg \text{eats}(y, z) \vee \text{killed}(y) \vee \text{food}(z)$
- $\text{eats}(\text{Anil}, \text{Peanuts})$
- $\text{alive}(\text{Anil})$
- $\neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$
- $\text{killed}(g) \vee \text{alive}(g)$
- $\neg \text{alive}(k) \vee \neg \text{killed}(k)$
- $\text{likes}(\text{John}, \text{Peanuts}).$

Note: Statements " $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$ " and " $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$ " can be written in two separate statements.

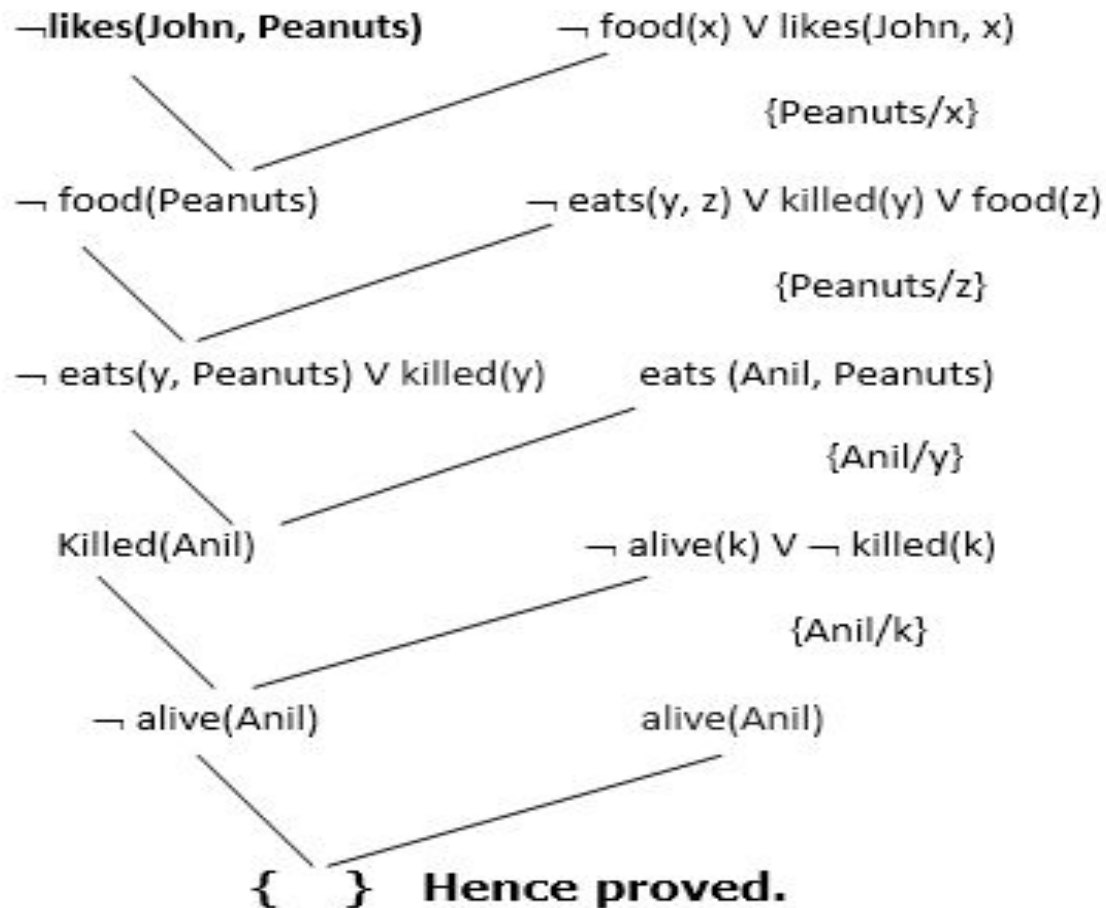
# Resolution in FOL

107

- **Step-3: Negate the statement to be proved**
- In this statement, we will apply negation to the conclusion statements, which will be written as  $\neg \text{likes}(\text{John}, \text{Peanuts})$
- **Step-4: Draw Resolution graph:**
- Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:

# Resolution in FOL

108



# Resolution in FOL

109

- Explanation of Resolution graph:
- In the first step of resolution graph,  $\neg \text{likes}(\text{John}, \text{Peanuts})$  , and  $\text{likes}(\text{John}, x)$  get resolved(canceled) by substitution of  $\{\text{Peanuts}/x\}$ , and we are left with  $\neg \text{food}(\text{Peanuts})$
- In the second step of the resolution graph,  $\neg \text{food}(\text{Peanuts})$  , and  $\text{food}(z)$  get resolved (canceled) by substitution of  $\{\text{Peanuts}/z\}$ , and we are left with  $\neg \text{eats}(y, \text{Peanuts}) \vee \text{killed}(y)$  .
- In the third step of the resolution graph,  $\neg \text{eats}(y, \text{Peanuts})$  and  $\text{eats}(\text{Anil}, \text{Peanuts})$  get resolved by substitution  $\{\text{Anil}/y\}$ , and we are left with  $\text{Killed}(\text{Anil})$  .
- In the fourth step of the resolution graph,  $\text{Killed}(\text{Anil})$  and  $\neg \text{killed}(k)$  get resolve by substitution  $\{\text{Anil}/k\}$ , and we are left with  $\neg \text{alive}(\text{Anil})$  .
- In the last step of the resolution graph  $\neg \text{alive}(\text{Anil})$  and  $\text{alive}(\text{Anil})$  get resolved.

# Forward chaining

110

- Inference engine:
- The inference engine is the component of the intelligent system in artificial intelligence, 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. Inference engine commonly proceeds in two modes, which are:
  - **Forward chaining**
  - **Backward chaining**

# Forward chaining

111

- Forward Chaining
- Forward chaining is also known as a forward deduction or forward reasoning method when using an inference engine. Forward chaining 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.
- The Forward-chaining algorithm 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.

# Forward chaining

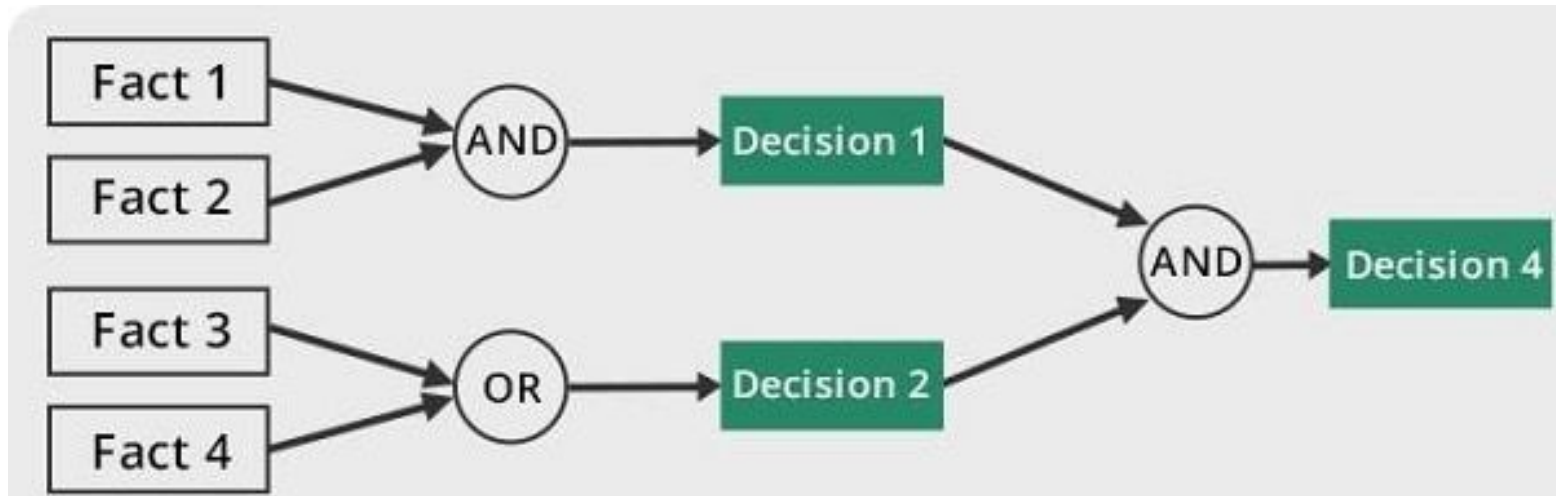
112

- **Properties of Forward-Chaining:**
- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as data-driven as we reach to the goal using available data.
- Forward -chaining approach is commonly used in the expert system, such as CLIPS, business, and production rule systems.



# Forward chaining

113



# Forward chaining

114

- Consider the following famous example which we will use in both approaches:  
Example:
- **"As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."**
- **Prove that "Robert is criminal."**
- To solve the above problem, first, we will convert all the above facts into first-order definite clauses, and then we will use a forward-chaining algorithm to reach the goal.

# Forward chaining

115

- It is a crime for an American to sell weapons to hostile nations. (Let's say  $p$ ,  $q$ , and  $r$  are variables)  
**American( $p$ )  $\wedge$  weapon( $q$ )  $\wedge$  sells( $p, q, r$ )  $\wedge$  hostile( $r$ )  $\rightarrow$  Criminal( $p$ ) ... (1)**
- Country A has some missiles.  **$\exists p$  Owns(A,  $p$ )  $\wedge$  Missile( $p$ )**. It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.  
**Owns(A, T1) ..... (2)**  
**Missile(T1) ..... (3)**

# Forward chaining

116

- All of the missiles were sold to country A by Robert.  
 $\forall \text{ Missiles}(p) \wedge \text{Owns}(\text{A}, p) \rightarrow \text{Sells}(\text{Robert}, p, \text{A}) \quad \text{.....(4)}$
- Missiles are weapons.  
 $\text{Missile}(p) \rightarrow \text{Weapons}(p) \quad \text{.....(5)}$
- Enemy of America is known as hostile.  
 $\text{Enemy}(p, \text{America}) \rightarrow \text{Hostile}(p) \quad \text{.....(6)}$
- Country A is an enemy of America.  
 $\text{Enemy}(\text{A}, \text{America}) \quad \text{.....(7)}$
- Robert is American  
 $\text{American}(\text{Robert}). \quad \text{.....(8)}$

# Forward chaining

117

- Forward chaining proof:
- **Step-1:**
- In the first step we will start with the known facts and will choose the sentences which do not have implications, such as: **American(Robert)**, **Enemy(A, America)**, **Owns(A, T1)**, and **Missile(T1)**. All these facts will be represented as below.

American (Robert)

Missile (T1)

Owns (A,T1)

Enemy (A, America)

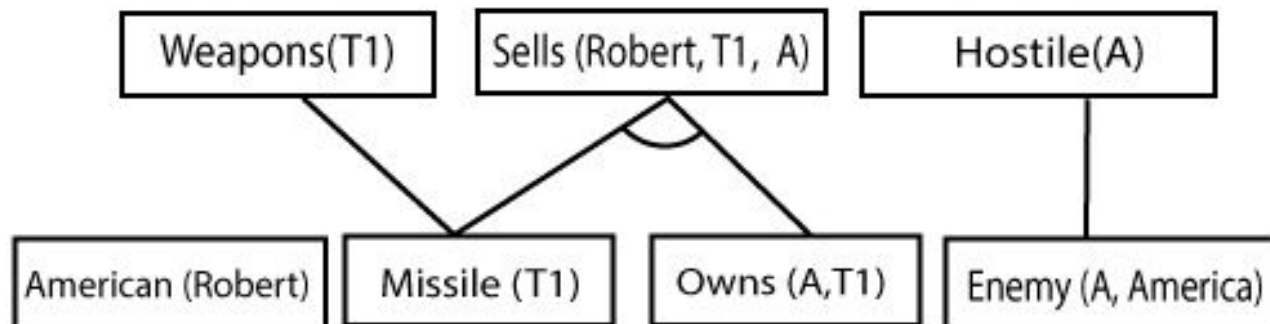
# Forward chaining

118

- **Step-2:**
- At the second step, we will see those facts which infer from available facts and with satisfied premises.
- Rule-(1) does not satisfy premises, so it will not be added in the first iteration.
- Rule-(2) and (3) are already added.
- Rule-(4) satisfy with the substitution  $\{p/T1\}$ , so **Sells (Robert, T1, A)** is added, which infers from the conjunction of Rule (2) and (3).
- Rule-(6) is satisfied with the substitution  $(p/A)$ , so **Hostile(A)** is added and which infers from Rule-(7).

# Forward chaining

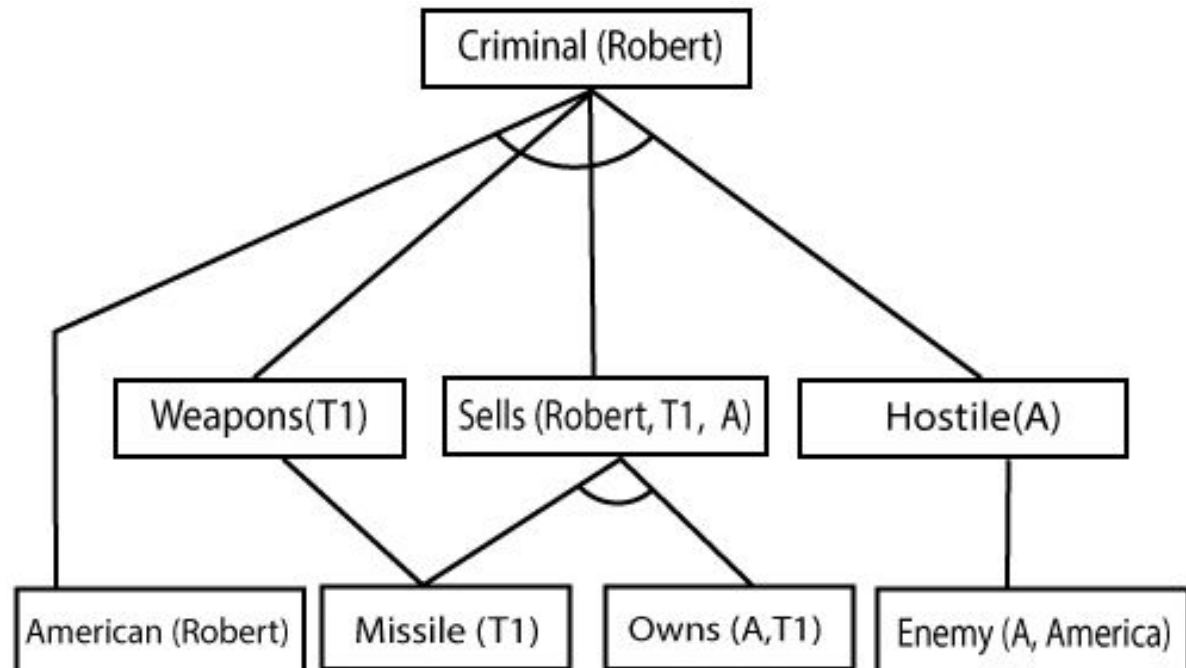
119



# Forward chaining

120

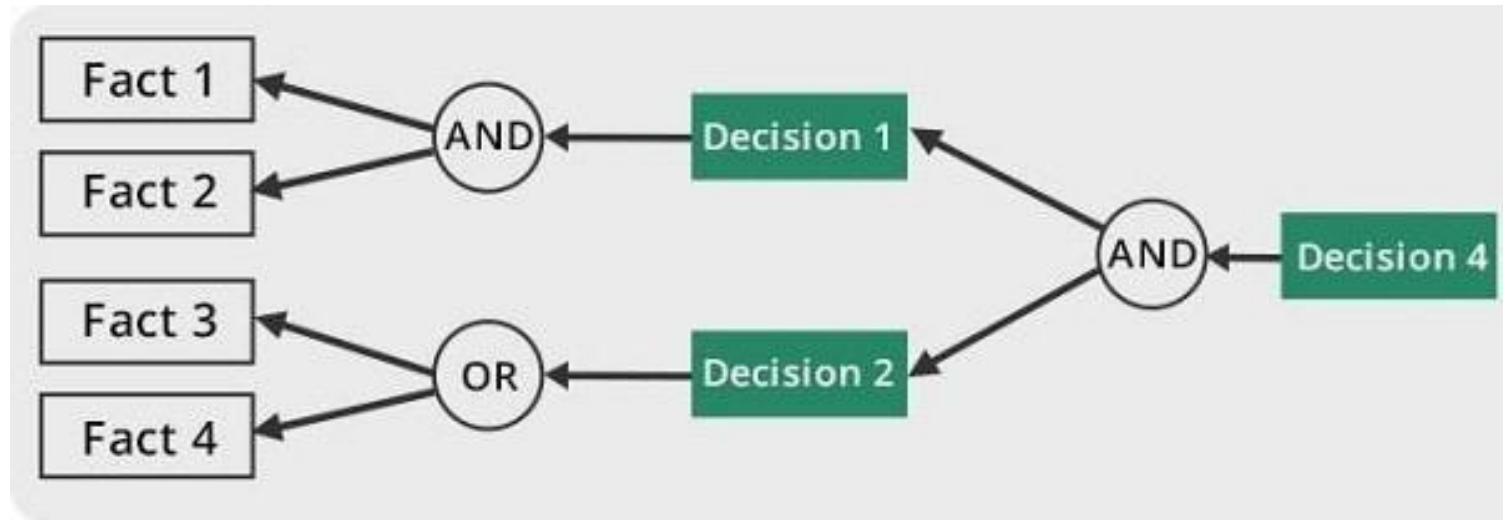
- **Step-3:**
- At step-3, as we can check Rule-(1) is satisfied with the substitution  $\{p/\text{Robert}, q/T1, r/A\}$ , so we can add **Criminal(Robert)** which infers all the available facts. And hence we reached our goal statement.





# Backward Chaining

121



# Backward Chaining

122

- Backward-chaining is also known as a backward deduction or backward reasoning method when using an inference engine. A backward chaining algorithm 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:**
  - It is known as a top-down approach.
  - Backward-chaining is based on modus ponens inference rule.
  - In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.

# Backward Chaining

123

- It is called a goal-driven approach, as a list of goals decides which rules are selected and used.
- Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.
- The backward-chaining method mostly used a **depth-first search** strategy for proof.

# Backward Chaining

124

- Example:
- In backward-chaining, we will use the same above example, and will rewrite all the rules.
- **$\text{American}(p) \wedge \text{weapon}(q) \wedge \text{sells}(p, q, r) \wedge \text{hostile}(r) \rightarrow \text{Criminal}(p) \dots(1)$**   
 **$\text{Owns}(A, T1) \dots\dots\dots(2)$**
- **$\text{Missile}(T1)$**
- **$?p \text{ Missiles}(p) \wedge \text{Owns}(A, p) \rightarrow \text{Sells}(\text{Robert}, p, A) \dots\dots(4)$**
- **$\text{Missile}(p) \rightarrow \text{Weapons}(p) \dots\dots\dots(5)$**
- **$\text{Enemy}(p, \text{America}) \rightarrow \text{Hostile}(p) \dots\dots\dots(6)$**
- **$\text{Enemy}(A, \text{America}) \dots\dots\dots(7)$**
- **$\text{American}(\text{Robert}). \dots\dots\dots(8)$**

# Backward Chaining

125

- Backward-Chaining proof:
- In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.
- **Step-1:**
- At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. So our goal fact is "Robert is Criminal," so following is the predicate of it.

Criminal (Robert)

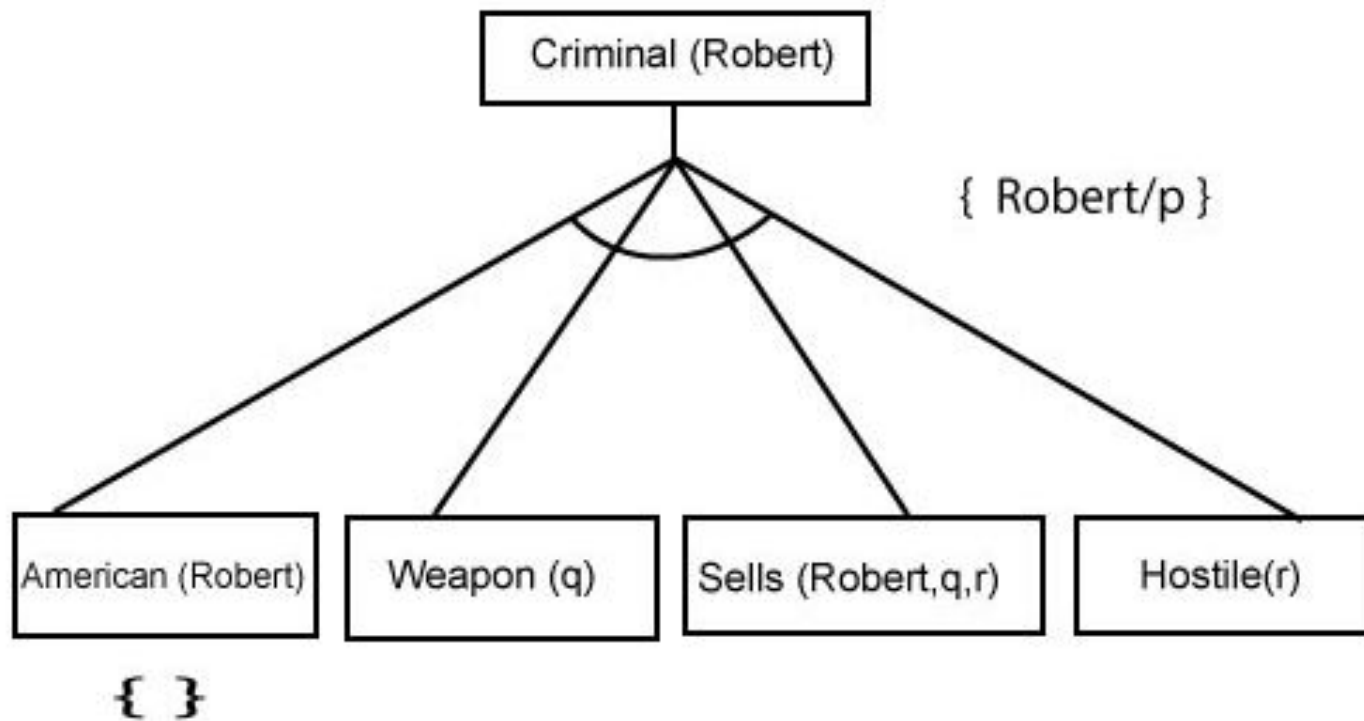
# Backward Chaining

126

- **Step-2:**
- At the second step, we will infer other facts from goal fact which satisfies the rules. So as we can see in Rule-1, the goal predicate Criminal (Robert) is present with substitution {Robert/P}. So we will add all the conjunctive facts below the first level and will replace p with Robert.
- **Here we can see American (Robert) is a fact, so it is proved here.**

# Backward Chaining

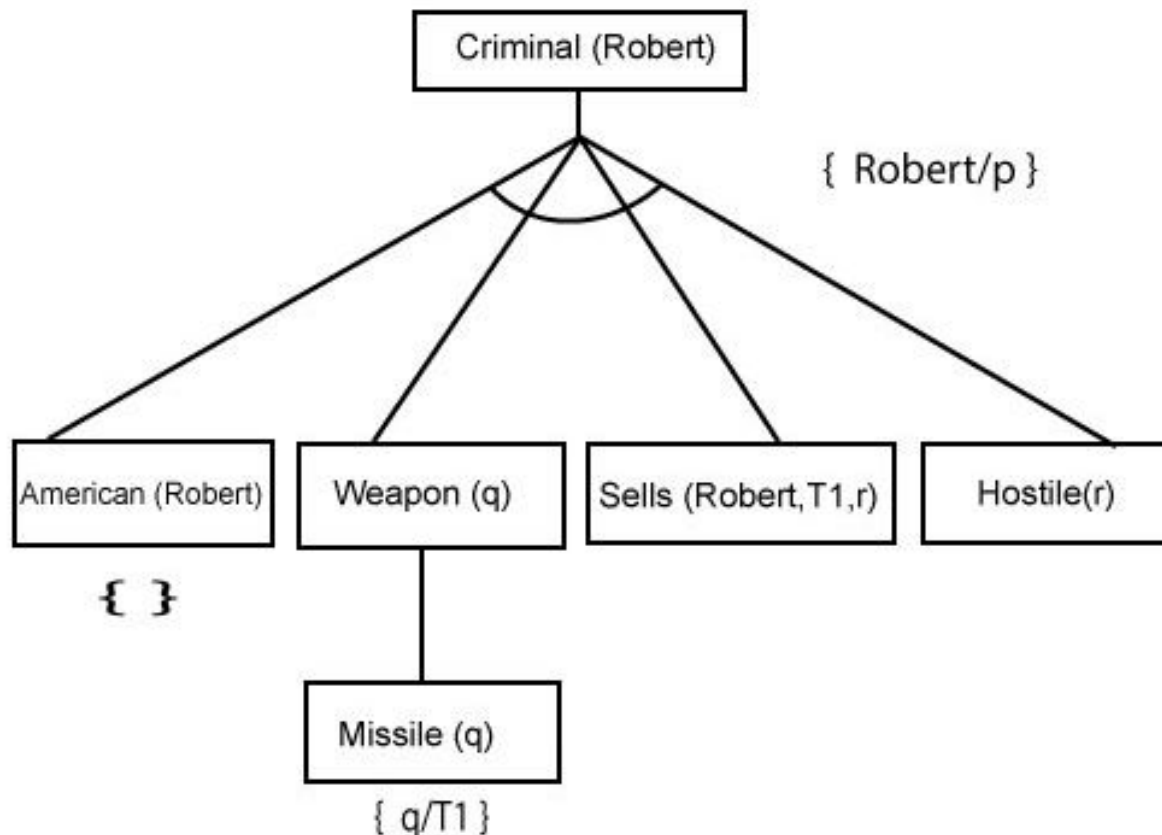
127



# Backward Chaining

128

- **Step-3:** At step-3, we will extract further fact  $\text{Missile}(q)$  which infer from  $\text{Weapon}(q)$ , as it satisfies Rule-(5).  $\text{Weapon}(q)$  is also true with the substitution of a constant  $T1$  at  $q$ .

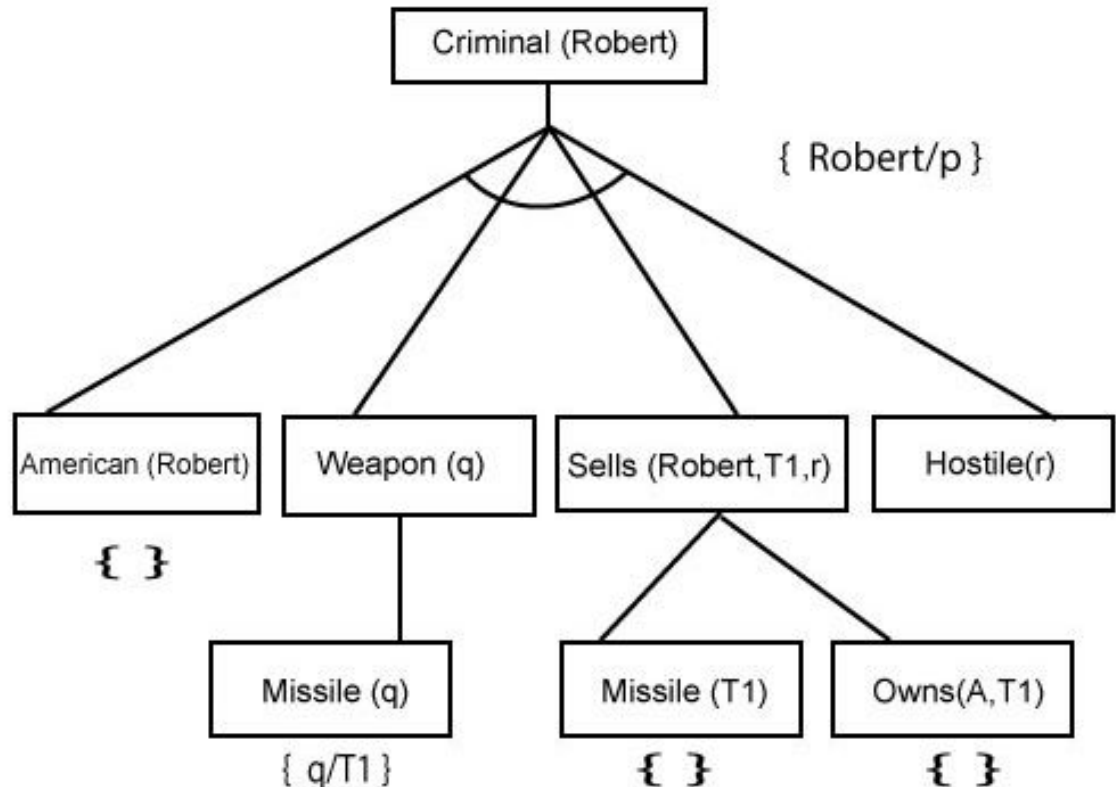




# Backward Chaining

129

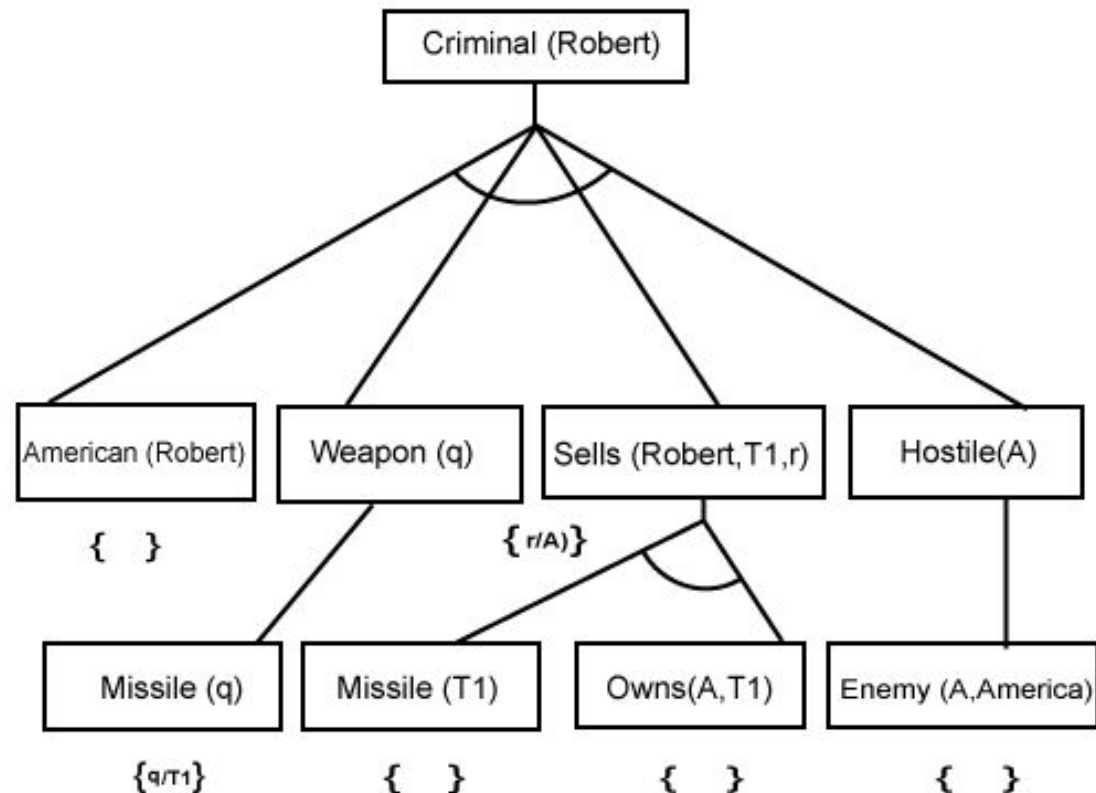
- **Step-4:**
- At step-4, we can infer facts  $\text{Missile}(T1)$  and  $\text{Owns}(A, T1)$  from  $\text{Sells}(\text{Robert}, T1, r)$  which satisfies the **Rule- 4**, with the substitution of  $A$  in place of  $r$ . So these two statements are proved here.



# Backward Chaining

130

- **Step-5:**
- At step-5, we can infer the fact **Enemy(A, America)** from **Hostile(A)** which satisfies Rule- 6. And hence all the statements are proved true using backward chaining.



# Backward Chaining

131

- 1) Gita loves all types of clothes.
- 2) Suits are clothes.
- 3) Jackets are clothes.
- 4) Anything any wear and isn't bad is clothes.
- 5) Sita wears skirt and is good.
- 6) Renu wears anything Sita wears.
- **Apply backward chaining and prove that Gita loves Kurtis.**

# Backward Chaining

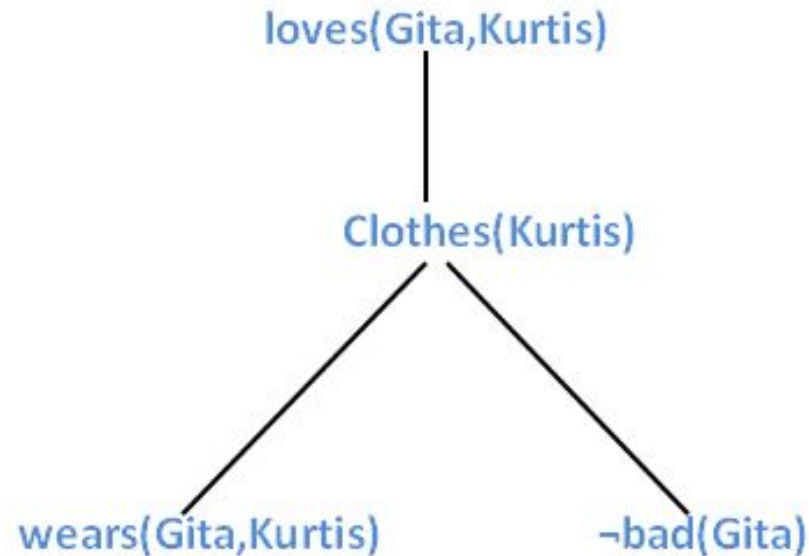
132

- **Solution:** Convert the given axioms into FOPL as:
- $x: \text{clothes}(x) \rightarrow \text{loves}(\text{Gita}, x).$
- $\text{Suits}(x) \rightarrow \text{Clothes}(x).$
- $\text{Jackets}(x) \rightarrow \text{Clothes}(x).$
- $\text{wears}(x, y) \rightarrow \neg \text{bad}(y) \rightarrow \text{Clothes}(x)$
- $\text{wears}(\text{Sita}, \text{skirt}) \rightarrow \neg \text{good}(\text{Sita})$
- $\text{wears}(\text{Sita}, x) \rightarrow \text{wears}(\text{Renu}, x)$
- **To prove:** Gita loves Kurtis.
- **FOPL:**  $\text{loves}(\text{Gita}, \text{Kurtis}).$

# Backward Chaining

133

- Apply backward chaining in the below graph:



- It is clear from the above graph Gita wears Kurtis and does not look bad. Hence, **Gita loves Kurtis.**

# Backward Chaining

134

- **Example :**

**<https://analyticssteps.com/blogs/forward-chaining-vs-backward-chaining>**