

## Unit:4 Knowledge Representation LH 8

Presented By : Tekendra Nath Yogi

[Tekendranath@gmail.com](mailto:Tekendranath@gmail.com)

College Of Applied Business And Technology

# Contd...

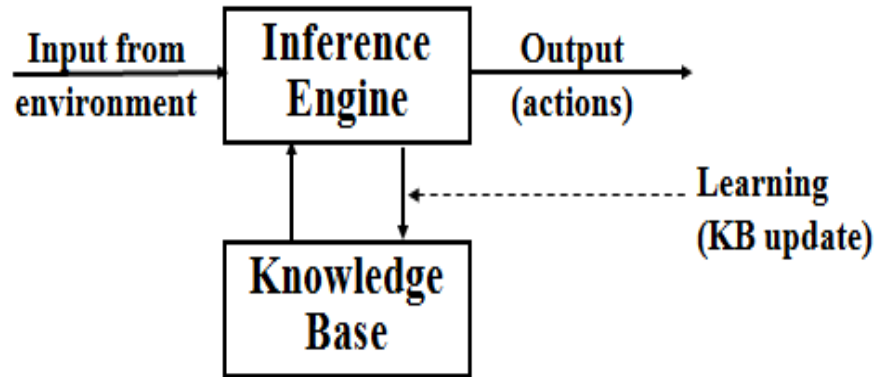
- **Outline:**
  - 4.1 Logic
    - 4.1.1 Propositional Logic
      - 4.1.1.1. Syntax, semantics, and properties
      - 4.1.1.2. Conjunctive Normal Form (CNF)
      - 4.1.1.3. Disjunctive Normal Form (DNF)
      - 4.1.1.4. Inference Rules
      - 4.1.1.5. Resolution
    - 4.1.2 Predicate Logic
      - 4.1.1.1. First-Order Predicate Logic (FOPL)
      - 4.1.1.2. Syntax and semantics in FOPL
      - 4.1.1.3. Quantifiers
      - 4.1.1.4. Clausal Normal Form
      - 4.1.1.5. Resolution
    - 4.1.3 Fuzzy Logics
  - 4.2 Semantic networks (nets): Introduction, and examples

# Knowledge

- Knowledge is a theoretical or practical understanding of a subject or a domain.
- Knowledge is also the sum of what is currently known.
- **Types of knowledge:**
  - Classification-based Knowledge :Ability to classify information
  - Decision-oriented Knowledge: Choosing the best option
  - Descriptive knowledge: State of some world (heuristic)
  - Procedural knowledge: How to do something
  - Reasoning knowledge: What conclusion is valid in what situation?
  - Assimilative knowledge: What its impact is?

# A Knowledge-Based Agent

- A knowledge-based agent consists of a knowledge base (KB) and an inference engine (IE).



- A **knowledge-base** is a set of sentences of what one knows about the world.
- The **Inference engine** derives new sentences from the input and KB.
- The agent operates as follows:
  1. It receives percepts from environment
  2. It computes what action it should perform (by IE and KB)
  3. It performs the chosen action.

# Contd...

- **Properties for Knowledge Representation Systems:**

- The following properties should be possessed by a knowledge representation system.
- **Representational Adequacy**
  - the ability to represent the required knowledge;
- **Inferential Adequacy**
  - the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;
- **Inferential Efficiency**
  - the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;
- **Acquisitional Efficiency**
  - the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

# Knowledge Representation

- The objective of knowledge representation is to express the knowledge about the world in a computer-tractable form.
- This computer- tractable form of knowledge helps the agent to identify patterns of good reasoning and patterns of bad reasoning, so the agent know which to follow and which to avoid.
- A formal language is required to represent knowledge in a computer tractable form and reasoning processes are required to manipulate this knowledge to deduce new facts.
- Key aspects of knowledge representation languages are:
  - **Syntax**: describes how sentences are formed in the language.
  - **Semantics**: describes the meaning of sentences, what is it the sentence refers to in the real world.
  - **Proof Theory(Inference method)**: Set of rules for generating new sentences that are necessarily true given that the old sentences are true.

# Contd....

- Knowledge Representation using Logic:
  - Logic is defined as a formal language for expressing knowledge and ways of reasoning.
  - Therefore, it should have syntax, semantics and inference method.
    - Syntax: describes how sentences are formed in the LOGIC.
    - Semantics: describes the meaning of sentences.
    - Inference method: set of rules for generating new sentences.

# Contd..

- Compared to natural languages (expressive but context sensitive) and programming languages (good for concrete data structures but not expressive) logic combines the advantages of natural languages and formal languages.
- So, Logic is:
  - Concise, unambiguous, context insensitive, expressive, effective for inferences
- Examples of Logics are:
  - Propositional logic
  - Predicate Logic and
  - Fuzzy Logic



# Propositional Logic

- Propositional logic is the simplest formal logic for the representation of the knowledge in terms of propositions.
  - **Proposition** is a declarative statement that is either true or false but not both.
  - If a proposition is true, then we say it has a **truth value** of "**true**"; if a proposition is false, its **truth value** is "**false**".
  - Some **examples** of Propositions are given below :
    - "Man is Mortal", it returns truth value "TRUE"
    - " $12 + 9 = 3 - 2$ ", it returns truth value "FALSE"
  - The following sentences are not Proposition:
    - "A is less than 2". It is because unless we give a specific value of A, we cannot say whether the statement is true or false.
    - Also the sentences "Close the door", and "Is it hot outside ?" are not propositions.

# Syntax of Propositional Logic

- Syntax of the propositional logic defines the:
  - Which symbols can be use (English: letters, punctuation)
  - Rules for constructing legal sentences in the logic.
  - How we are allowed to combine symbols
- Symbols:
  - Logical constants: true, false
  - Propositional symbols: P, Q, R, S, ... , etc
  - Wrapping parentheses: ( ... )

# Contd...

- **Atomic formulas(Sentence)**: Propositional Symbols or logical constants.
- **Literals**: atomic sentences and their negations
- **Complex Formulas**: can be formed by combining atomic formulas with the following **connectives**:

$\neg$  ...not [negation]

$\wedge$  ...and [conjunction]

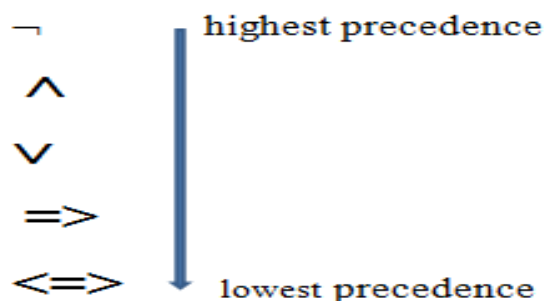
$\vee$  ...or [disjunction]

$\longrightarrow$  ...implies [implication / conditional]

$\longleftrightarrow$  ..is equivalent [biconditional]

# Contd..

- A **sentence** is defined as follows:
  - A symbol is a sentence
  - If  $S$  is a sentence, then  $\neg S$  is a sentence
  - If  $S$  is a sentence, then  $(S)$  is a sentence
  - If  $S$  and  $T$  are sentences, then  $(S \vee T)$ ,  $(S \wedge T)$ ,  $(S \rightarrow T)$ , and  $(S \leftrightarrow T)$  are sentences
  - A sentence results from a finite number of applications of the above rules
- **Order of precedence** of logical connectors



e.g.  $\neg P \vee Q \wedge R \Rightarrow S$  is equivalent to  $((\neg P) \vee (Q \wedge R)) \Rightarrow S$

# Contd..

- A BNF (Backus–Naur Form) grammar of sentences in propositional logic is shown in below:

Sentence	$\rightarrow$ Atomic Sentence   Complex _Sentence
Atomic _Sentence	$\rightarrow$ True   False   Symbol
Symbol	$\rightarrow$ P   Q   R .....
Complex _Sentence	$\rightarrow$ $\neg$ Sentence   (Sentence $\wedge$ Sentence)   (Sentence $\vee$ Sentence)   (Sentence $\Rightarrow$ Sentence)   (Sentence $\Leftrightarrow$ Sentence)

# Examples of PL sentences

- P means "It is hot"
- Q means "It is humid"
- R means "It is raining"
- $P \wedge Q \Rightarrow R$

"If it is hot and humid, then it is raining"

- $Q \Rightarrow P$

"If it is humid, then it is hot"

- Q

"It is humid."

# Propositional Logic (PL): Semantics

- The semantics of propositional logic defines the rules for determining the truth of a sentence with respect to a particular given model (model fixes the truth value (true or false) for every proposition symbol).
  - **For example**, If the sentences in the knowledge base make use of the proposition symbols P, Q, and R then one possible model is:

$$\text{Model (m)} = \{ P = F, Q = F, R = T \}$$

- In propositional logic all sentences are constructed from atomic sentences and the five connectives
- So semantics of propositional logic requires rules to define the truth of atomic sentences and rules to define the truth of sentences formed with each of the five connectives

# Contd..

- How to compute the truth of atomic sentences?
  - The truth value of Atomic sentences can be determined by using the following rules :
    - True is true in every model and False is false in every model.
    - The truth value of every other proposition symbol must be specified directly in the model.
    - For example, in the model  $m$  given earlier,  $P$  is false(F).



# Contd...

- How to compute the truth of sentences formed with each of the five connectives?
  - For complex sentences, we have five rules, which hold for any sub-sentences  $P$  and  $Q$  in any model  $m$ :

$\neg P$  is true iff  $P$  is false in  $m$ .

$P \wedge Q$  is true iff both  $P$  and  $Q$  are true in  $m$ .

$P \vee Q$  is true iff either  $P$  or  $Q$  is true in  $m$ .

$P \Rightarrow Q$  is true unless  $P$  is true and  $Q$  is false in  $m$ .

$P \Leftrightarrow Q$  is true iff  $P$  and  $Q$  are both true or both false in  $m$ .

- The above rules can be summarized as follows:

$P$	$Q$	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

- From these tables, the truth value of any sentence  $s$  can be computed with respect to any model  $m$  by a simple recursive evaluation.

# Properties

- Validity(Tautology)
- Satisfiability (contingency)
- Un-Satisfiability (Contradictory)
- Equivalent
- Entailment
- Completeness
- Soundness

# Contd..

- **Validity:**
  - A sentence is valid if it is true in all models,  
e.g.,  $\text{True}$ ,  $A \vee \neg A$ ,  $A \Rightarrow A$ ,  $(A \wedge (A \Rightarrow B)) \Rightarrow B$
  - Valid sentences are also known as **tautologies**. Every valid sentence is logically equivalent to True
- **Example :** Prove  $[(A \rightarrow B) \wedge A] \rightarrow B$  is a tautology

**Solution:** The truth table is as follows

A	B	$A \rightarrow B$	$(A \rightarrow B) \wedge A$	$[(A \rightarrow B) \wedge A] \rightarrow B$
True	True	True	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	True

As we can see every value of  $[(A \rightarrow B) \wedge A] \rightarrow B$  is "True", it is a tautology.

# Contd..

- **Satisfiability:**
  - A sentence is satisfiable if it is true in some model
    - E.g.,  $A \vee A$
  - Satisfiable sentences are also known as **Contingency**.
- **Example:** Prove  $(A \vee B) \wedge (\neg A)$  a contingency

**Solution:** The truth table is as follows

A	B	$A \vee B$	$\neg A$	$(A \vee B) \wedge (\neg A)$
True	True	True	False	False
True	False	True	False	False
False	True	True	True	True
False	False	False	True	False

As we can see every value of  $(A \vee B) \wedge (\neg A)$  has both “True” and “False”, it is a contingency.

# Contd..

- **Un-Satisfiability (Contradictory) :**
  - A sentence is un-satisfiable if it is true in no models
    - E.g.,  $A \wedge \neg A$
  - Un-Satisfiable sentences are also known as contradictory sentences.
- **Example:** Prove  $(A \vee B) \wedge (\neg A) \wedge (\neg B)$  is a contradiction
- Solution:** The truth table is as follows :

A	B	$A \vee B$	$\neg A$	$\neg B$	$(\neg A) \wedge (\neg B)$	$(A \vee B) \wedge [(\neg A) \wedge (\neg B)]$
True	True	True	False	False	False	False
True	False	True	False	True	False	False
False	True	True	True	False	False	False
False	False	False	True	True	True	False

As we can see every value of  $(A \vee B) \wedge (\neg A) \wedge (\neg B)$  is “False”, it is a contradiction.

# Contd..

- **Propositional Equivalences**
- Two statements X and Y are logically equivalent if any of the following two conditions hold :
  - The truth tables of each statement have the same truth values.
  - The bi-conditional statement  $X \Leftrightarrow Y$  is a tautology.
- **Example :** Prove  $\neg(A \vee B)$  and  $[(\neg A) \wedge (\neg B)]$  are equivalent  
Testing by 1<sup>st</sup> method (Matching truth table)

A	B	$A \vee B$	$\neg (A \vee B)$	$\neg A$	$\neg B$	$[(\neg A) \wedge (\neg B)]$
True	True	True	False	False	False	False
True	False	True	False	False	True	False
False	True	True	False	True	False	False
False	False	False	True	True	True	True

Here, we can see the truth values of  $\neg(A \vee B)$  and  $[(\neg A) \wedge (\neg B)]$  are same, hence the statements are equivalent.

# Contd..

- Testing by 2<sup>nd</sup> method (Bi-conditionality)

A	B	$\neg (A \vee B)$	$[(\neg A) \wedge (\neg B)]$	$[\neg (A \vee B)] \Leftrightarrow [(\neg A) \wedge (\neg B)]$
True	True	False	False	True
True	False	False	False	True
False	True	False	False	True
False	False	True	True	True

- As  $[\neg(A \vee B)] \Leftrightarrow [(\neg A) \wedge (\neg B)]$  is a tautology, the statements are equivalent.

# Contd..

- **Standard logical equivalences:**

- Some standard logical equivalences in propositional logic is as shown below:
- The symbols  $\alpha$ ,  $\beta$ , and  $\gamma$  stand for arbitrary sentences of propositional logic.

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$

$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$

$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{De Morgan}$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{De Morgan}$$

$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$

$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$



# Contd..

- **Entailment:**
  - Entailment means that one thing follows from another:
    - E.g .,  $x + y = 4$  entails  $4 = x + y$
    - KB as a haystack and  $\alpha$  as a needle. Entailment is like the needle being in the haystack; inference is like finding it.
  - Entailment is a relationship between sentences that is based on semantics
  - Knowledge base KB entails sentence  $\alpha$  if and only if  $\alpha$  is true in all worlds where KB is true . :  $KB \models \alpha$
  - If an inference algorithm  $i$  can derive  $\alpha$  from KB, we write:  $KB \vdash_i \alpha$
  - which is pronounced “ $\alpha$  is derived from KB by  $i$ ” or “ $i$  derives  $\alpha$  from KB.”

# Contd..

- **Soundness:**

- An inference algorithm that derives only entailed sentences is called sound or truth preserving. Soundness is a highly desirable property.
- An unsound inference procedure essentially makes things up as it goes along—it announces the discovery of nonexistent needles.

# Contd..

- **Completeness:**

- An inference algorithm is complete if it can definitely derive any sentence that is entailed.
- For real haystacks, which are finite in extent, it seems obvious that a systematic examination can always decide whether the needle is in the haystack.
- For many knowledge bases, however, the haystack of consequences is infinite, and completeness becomes an important issue.
- Fortunately, there are complete inference procedures for logics that are sufficiently expressive to handle many knowledge bases.

# Inference Methods

- The process by which a conclusion is drawn from given premises.
- But logic is concerned with: does the truth of the conclusion follow from that of the premises?
- Several basic methods for determining whether a given set of premises propositionally entails a given conclusion.
  - Truth Table Method(Enumeration Method)
  - Deductive (Proof) Systems
  - Resolution

# Contd..

- Enumeration Method:

- Example1:

- Let, conclusion  $\alpha = A \vee B$  and  $KB = (A \vee C) \wedge (B \vee \neg C)$
  - Does  $KB$  entail  $\alpha$ ?
    - check all possible models;  $\alpha$  must be true whenever  $KB$  is true i.e.,  $KB \rightarrow \alpha$  is valid.

$A$	$B$	$C$	$A \vee C$	$(B \vee \neg C)$	$KB$	$\alpha$
True	True	True	True	True	True	True
True	True	False	True	True	True	True
True	False	True	True	False	False	True
True	False	False	True	True	True	True
False	True	True	True	True	True	True
False	True	False	False	True	False	True
False	False	True	True	False	False	False
False	False	False	False	True	False	False

So, conclusion

KB entails  $\alpha$

# Contd..

- Example2:

- Symbols:

- P is “It is hot”,
    - Q is “It is humid” and
    - R is “It is raining”.

- KB:

- $P \wedge Q \Rightarrow R$  (“If it is hot and humid, then it is raining”),
    - $Q \Rightarrow P$  (“If it is humid, then it is hot”),
    - Q (“It is humid”).

- Question:

- Is it raining? (i.e., is R entailed by KB?)

$P, Q, R$	$P \wedge Q \Rightarrow R$	$Q \Rightarrow P$	$Q$	$KB$	$R$	$KB \Rightarrow R$
$T, T, T$	$T$	$T$	$T$	$T$	$T$	$T$
$T, T, F$	$F$	$T$	$T$	$F$	$F$	$T$
$T, F, T$	$T$	$T$	$F$	$F$	$T$	$T$
$T, F, F$	$T$	$T$	$F$	$F$	$F$	$T$
$F, T, T$	$T$	$F$	$T$	$F$	$T$	$T$
$F, T, F$	$T$	$F$	$T$	$F$	$F$	$T$
$F, F, T$	$T$	$T$	$F$	$F$	$T$	$T$
$F, F, F$	$T$	$T$	$F$	$F$	$F$	$T$

- So, R is entailed by the KB and we can conclude it is raining.

# Deductive (Proof) Systems

- Done by applying rules of inference directly to the sentences in our knowledge base to construct a proof of the desired sentence without consulting models.
  - **Example:**
    - All men are mortal.
    - Ram is a man.
    - Therefore, Ram is mortal.
- If the number of models is large but the length of the proof is short, then Deductive proof can be more efficient than model checking.

# Inference Rules for Propositional Logic

- Logically equivalences are used as a inference rules and additional inference rules are:

- (MP) Modes Ponens (Implication-elimination):  $\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$

- (AI) And-introduction (OI) Or-introduction

$$\frac{\alpha_1, \alpha_2, \dots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}$$

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_n}$$

- (AE) And-elimination:

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}{\alpha_i}$$

- (NE) Negation-elimination

$$\frac{\neg \neg \neg \alpha}{\alpha}$$





# Contd...

- **Example:**
- **Symbols:**
  - P is “It is hot”,
  - Q is “It is humid” and
  - R is “It is raining”.
- **KB:**
  - $P \wedge Q \Rightarrow R$  (“If it is hot and humid, then it is raining”),
  - $Q \Rightarrow P$  (“If it is humid, then it is hot”),
  - Q (“It is humid”).
- **Question:**
  - Is it raining? (i.e., is R entailed by KB?)

Step		Reason
1	$Q$	(premise)
2	$Q \Rightarrow P$	(premise)
3	$P$	(modus ponens) (1,2)
4	$(P \wedge Q) \Rightarrow R$	(premise)
5	$P \wedge Q$	(and-intro) (1,3)
6	$R$	(and-elim) (4,5)

- So, R is entailed by the KB and we can conclude it is raining.

# Normal Forms

- There are two major normal forms of statements in propositional logic. They are :
- Conjunctive Normal Form (CNF)
  - conjunction of disjunction of literals  *clauses*
  - E.g.,  $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
- Disjunctive Normal Form (DNF)
  - disjunction of conjunction of literals  *terms*
  - E.g.,  $(A \wedge \neg B) \vee (B \wedge \neg C) \vee (\neg C \wedge D \wedge A)$

# Conversion to CNF

- A sentence that is expressed as a conjunction of disjunctions of literals is said to be in conjunctive normal form (CNF). A sentence in CNF that contains only  $k$  literals per clause is said to be in  $k$ -CNF.
- *Algorithm:*
  - Eliminate  $\leftrightarrow$     rewriting  $P \leftrightarrow Q$  as  $(P \rightarrow Q) \wedge (Q \rightarrow P)$
  - Eliminate  $\rightarrow$     rewriting  $P \rightarrow Q$  as  $\neg P \vee Q$
  - Use De Morgan's laws to push  $\neg$  inwards:
    - rewrite  $\neg(P \wedge Q)$  as  $\neg P \vee \neg Q$
    - rewrite  $\neg(P \vee Q)$  as  $\neg P \wedge \neg Q$
  - Eliminate double negations: rewrite  $\neg \neg P$  as  $P$
  - Use the distributive laws to get CNF:
    - rewrite  $(P \wedge Q) \vee R$  as  $(P \vee R) \wedge (Q \vee R)$
  - Flatten nested clauses:
    - $(P \wedge Q) \wedge R$  as  $P \wedge Q \wedge R$
    - $(P \vee Q) \vee R$  as  $P \vee Q \vee R$

# Contd..

- Let's illustrate the conversion to CNF by using an example.

$$B \Leftrightarrow (A \vee C)$$

- Eliminate  $\Leftrightarrow$ , replacing  $\alpha \Leftrightarrow \beta$  with  $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ .
  - $(B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B)$
- Eliminate  $\Rightarrow$ , replacing  $\alpha \Rightarrow \beta$  with  $\neg \alpha \vee \beta$ .
  - $(\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B)$
- Move  $\neg$  inwards using de Morgan's rules and double-negation:
  - $(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B)$
- Apply distributivity law ( $\wedge$  over  $\vee$ ) and flatten:
  - $(\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B)$

# Proof by Resolution

- **Resolution Rules:**

- **(UR) Unit Resolution:** Unit resolution rule takes a clause (a disjunction of literals) and a literal and produces a new clause. Single literal is also called unit clause.

$$\frac{\ell_1 \vee \dots \vee \ell_k, \quad m}{\ell_1 \vee \dots \vee \ell_{i-1} \vee \ell_{i+1} \vee \dots \vee \ell_k}$$

- Where  $\ell_i$  and  $m$  are complementary literals

- For Example: 
$$\frac{\alpha \vee \beta, \quad \neg\beta}{\alpha}$$

# Contd..

- **General Resolution:** The unit resolution rule can be generalized to the full resolution rule,

$$\frac{\ell_1 \vee \dots \vee \ell_k, \quad m_1 \vee \dots \vee m_n}{\ell_1 \vee \dots \vee \ell_{i-1} \vee \ell_{i+1} \vee \dots \vee \ell_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

where  $\ell_i$  and  $m_j$  are complementary literals.

- **For Example:**

$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

# Proof by resolution

- Resolution is used with knowledge bases in CNF (or clausal form), and is complete for propositional logic.
- Resolution takes two clauses and produces a new clause containing all the literals of the two original clauses except the two complementary literals.
- In addition the resulting clause should contain only one copy of each literal. The removal of multiple copies of literals is called **factoring**.
- For example, if we resolve  $(A \vee B)$  with  $(A \vee \neg B)$ , we obtain  $(A \vee A)$ , which is reduced to just  $A$ .

# Contd..

- **Resolution Algorithm:**
  - Convert KB into CNF
  - Add negation of sentence to be entailed into KB i.e.  $(KB \wedge \neg\alpha)$
  - Then apply resolution rule to resulting clauses.
  - The process continues until:
    - There are no new clauses that can be added  
Hence *KB* **does not** entail  $\alpha$
    - Two clauses resolve to entail the empty clause.  
Hence *KB* **does** entail  $\alpha$



# Resolution Example1

**Example:** Consider the knowledge base given as:  $KB = (B \Leftrightarrow (A \vee C)) \wedge \neg B$   
Prove that  $\neg A$  can be inferred from above KB by using resolution.

Solution:

*At first, convert KB into CNF*

$$B \Rightarrow (A \vee C) \wedge ((A \vee C) \Rightarrow B) \wedge \neg B$$

$$(\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B) \wedge \neg B$$

$$(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B) \wedge \neg B$$

$$(\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B) \wedge \neg B$$

*Add negation of sentence to be inferred from KB into KB*

Now KB contains following sentences all in CNF

$$(\neg B \vee A \vee C)$$

$$(\neg A \vee B)$$

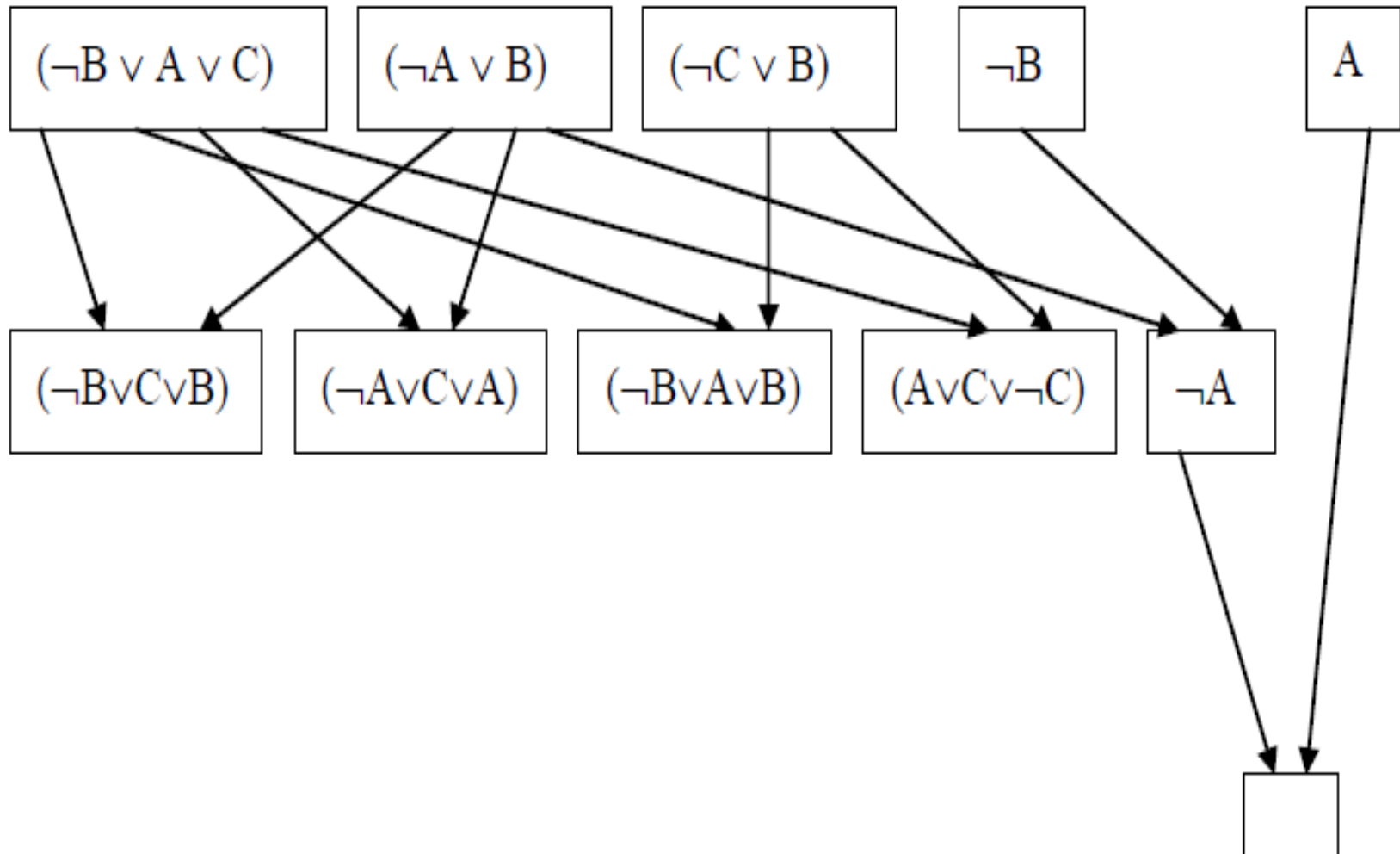
$$(\neg C \vee B)$$

$$\neg B$$

A (negation of conclusion to be proved)

*Now use Resolution algorithm*

# Contd..



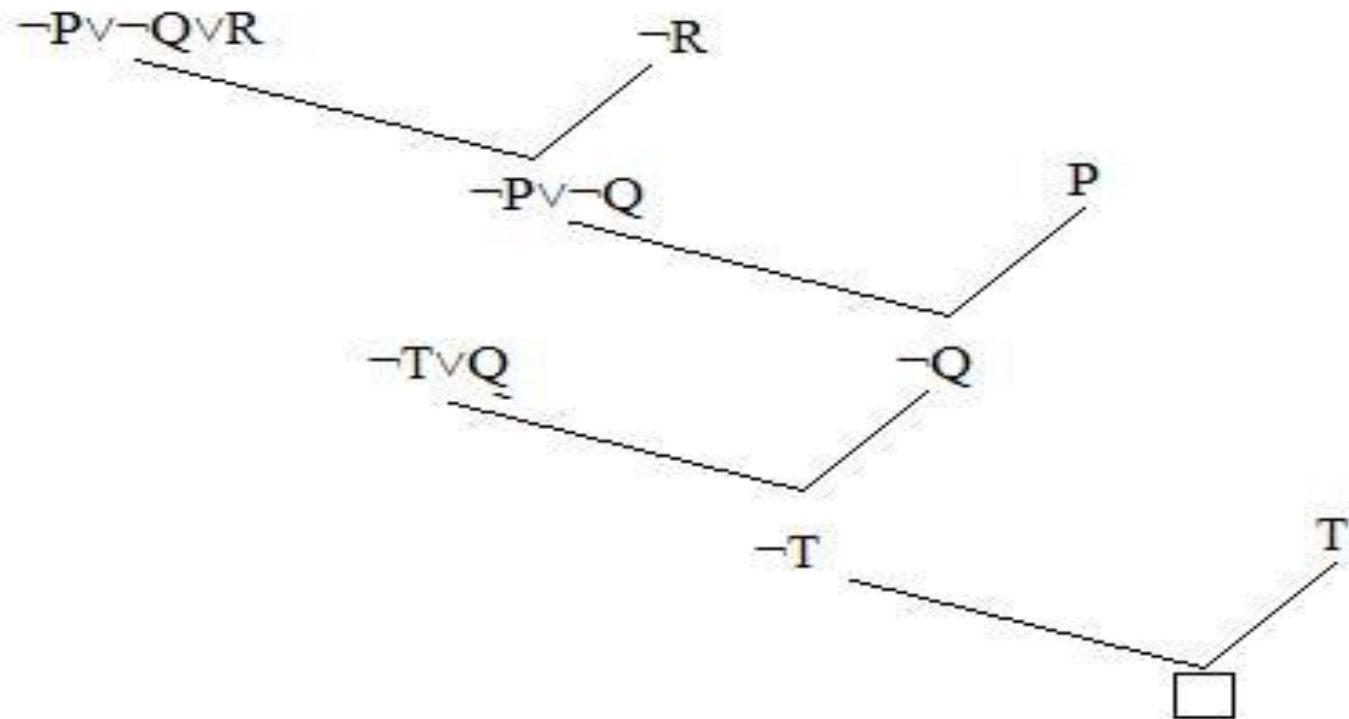
# Resolution Example2

- Suppose we are given the axioms shown in the 2nd column of Table below and we want to prove **R**.
- First we convert the axioms to clause which is already in clause form.

Sl. No	Given Axioms	Converted to Clause Form
1	P	P
2	$(P \wedge Q) \rightarrow R$	$\neg P \vee \neg Q \vee R$
3	$(S \vee T) \rightarrow Q$	$\neg S \vee \neg Q$
4		$\neg T \vee Q$
5	T	T

# Contd..

- Then we begin selecting pairs of clauses to resolve together.



# Pros and cons of propositional logic

- Propositional logic is declarative
- Propositional logic is compositional:
  - meaning of  $B \wedge P$  is derived from meaning of  $B$  and of  $P$
- Meaning in propositional logic is context-independent
  - (unlike natural language, where meaning depends on context)
- Propositional logic has very limited expressive power (unlike natural language)

# Comparison Between propositional logic and FOPL

- Propositional logic assumes the world contains facts, whereas first-order logic (like natural language) assumes the world contains:
  - Objects: people, houses, numbers, colors, baseball games, wars, ...
  - Relations: red, round, prime, brother of, bigger than, part of, comes between,...
  - Functions: father of, best friend, one more than, plus, ...
- The primary difference between PL and FOPL is their ontological commitment(**What** exists in the world — TRUTH)
  - PL: facts hold or do not hold.
  - FL : objects with relations between them that hold or do not hold



# Predicate Logic

Presented By : Tekendra Nath Yogi

[Tekendranath@gmail.com](mailto:Tekendranath@gmail.com)

College Of Applied Business And Technology



# Predicate Logic

- propositional logic is best to illustrate the basic concepts of logic and knowledge-based agents.
- But, Propositional logic is limited in several ways.
  - Hard to represent information concisely.
  - Must deal with facts that are either **TRUE** or **FALSE**.
- **Predicate Logic!** a more powerful logic (use foundation of propositional logic) by adding more expressive concepts.



# Contd..

- First-order logic (FOL) models the world in terms of
  - **Objects**, which are things with individual identities
  - **Properties** of objects that distinguish them from others
  - **Relations** that hold among sets of objects
  - **Functions**, which are a subset of relations where there is only one “value” for any given “input”
- **Examples:**
  - Objects: Students, lectures, companies, cars ...
  - Properties: blue, oval, even, large, ...
  - Relations: Brother, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
  - Functions: father-of, best-friend, second-half, one more-than ...

# Representing knowledge in FOPL

- The basic syntactic elements of first order logics are the symbols.
- Formula in FOPL contains two types of Symbols. They are:

## 1. User defined symbols

- Constants:
  - 3, John
  - Individuals
- Functions:
  - f,g,h
  - mappings
- Predicates:
  - $P(x,y)$
  - functions whose range is {True,False}

## 2. Logic defined symbols

- Variables:  
x,y,z  
Can be instantiated
- Logical Operators()
- Truth Symbols (TRUE, FALSE)

### Quantifiers:

- $\forall$  for all
- $\exists$  there exists

# User defined symbols

- **Constant Symbols :**
  - The objects from the real world are represented by constant symbols.
  - For instance, the symbol “Ram” may represent a certain individual called Ram.
  - E.g., Hari, Book, Three, etc.

# Contd..

- **Function Symbols:**

- Functions are a subset of the relations in which there is only one "value" for any given "input".
- i.e., Function symbols denote mappings from elements of a domain to elements of a domain.
- E.g., FatherOf(Ram), Cosine(0), etc.

# Contd..

- **Predicate Symbols:**

- **Properties of objects** are represented by predicates applied to those objects ( $P(a), \dots$ ): e.g. "male(Ram)" represents that Ram is a male.
- **Relationships between objects** are represented by predicates with more arguments: e.g., "Brother(Ram, Hari)" represents the fact that Ram is the brother of Hari.
- **The value of a predicate is one of the Boolean constants T (i.e. true) or F (i.e. false).** "brother(Ram, Hari) = T" means that the sentence "Ram is the brother of Hari" is true. "Brother(Tom, Bob) = F" means that the sentence "Ram is the Brother of Hari" is false.
- Besides constants, the arguments of the predicates may be functions (f, g,...) or variables (x, y,...).
- E.g. likes(john, mary), valuable(gold), Brother(father(Ram), Krishna), greter-

# Logic defined symbols

- First order logic provides some symbols on its own:
  - Truth Symbols : that are TRUE and FALSE.
  - Quantifiers:
    - Universal Quantifier ( $\forall$ ): means “for all”.
    - Existential quantification ( $\exists$ ): means “there exists”.
  - Logical Operators: order of precedence is left to right but the quantifiers have the same precedence as NOT.
    - $( )$ ,  $\neg$ ,  $\forall$ ,  $\exists$ ,  $\wedge$ ,  $\vee$ ,  $\Rightarrow$ ,  $\Leftrightarrow$

# Contd..

- Variable Symbols:
  - Variable symbols represent potentially any element of a domain and allow the formulation of general statements about the elements of the domain.
  - E.g.,  $x, y, z, \dots$  Etc.

# Sentences in first order logic

- Sentences in FOPL are built from **terms** and **predicates**:
  - A Term is an expression referring to an object; Therefore Terms can be:
    - constant symbols,
    - function symbols and
    - variable symbols .
    - A term with no variables is a **ground term** (
      - » E.g., john, father\_of(john) and etc.
  - Predicate Symbols for referring to relations
    - e.g., green(grass)



# Sentences in first order logic

- A sentence in FOL is either an **atomic** sentence or **complex** sentence.
- **Atom (Atomic Sentence):**
  - an n-ary predicate symbols that has value TRUE or FALSE.
  - Each of the n-ary arguments are terms.
  - predicate ( $\text{term}_1, \text{term}_2, \dots, \text{term}_k$ )where , term = function( $\text{term}_1, \text{term}_2, \dots, \text{term}_k$ ) or constant, or variable  
E,g., father(Ram, Hari), greater\_than(x,y) and etc...

# Sentences in first order logic

- **Complex Sentence:**

- A complex sentence is defined as a bunch of atomic sentences joined together using logical connectives (like AND, OR, IMPLICATION, BICONDITIONAL, NOT).

- »  $\neg S$ ,

- »  $S1 \wedge S2$ ,

- »  $S1 \vee S2$ ,

- »  $S1 \Rightarrow S2$ ,

- »  $S1 \Leftrightarrow S2$

- » Quantifier  $\langle \text{Variable} \rangle \langle \text{Sentence} \rangle$

# Predicate Logic Syntax

<i>Sentence</i>	→	<i>AtomicSentence</i>   ( <i>Sentence</i> <i>Connective</i> <i>Sentence</i> )   <i>Quantifier</i> <i>Variable</i> , ... <i>Sentence</i>   $\neg$ <i>Sentence</i>
<i>AtomicSentence</i>	→	<i>Predicate</i> ( <i>Term</i> , ...)   <i>Term</i> = <i>Term</i>
<i>Term</i>	→	<i>Function</i> ( <i>Term</i> , ...)   <i>Constant</i>   <i>Variable</i>
<i>Connective</i>	→	$\wedge$   $\vee$   $\Rightarrow$   $\Leftrightarrow$
<i>Quantifier</i>	→	$\forall$   $\exists$
<i>Constant</i>	→	<i>A</i> , <i>B</i> , <i>C</i> , <i>X</i> <sub>1</sub> , <i>X</i> <sub>2</sub> , <i>Jim</i> , <i>Jack</i>
<i>Variable</i>	→	<i>a</i> , <i>b</i> , <i>c</i> , <i>x</i> <sub>1</sub> , <i>x</i> <sub>2</sub> , <i>counter</i> , <i>position</i> , ...
<i>Predicate</i>	→	<i>Adjacent-To</i> , <i>Younger-Than</i> , <i>HasColor</i> , ...
<i>Function</i>	→	<i>Father-Of</i> , <i>Square-Position</i> , <i>Sqrt</i> , <i>Cosine</i>

ambiguities are resolved through precedence or parentheses

# Quantifiers in first order logic

- FOPL also provides variable quantifiers that allow the expression of properties for entire collections of objects instead of enumerating objects by name.
- Two types of quantifiers:
  - **Universal quantifier ( $\forall$ )**: means “for all”.
  - **Existential quantification ( $\exists$ )**: means “there exists”.
- Quantifiers are used with sentences containing variable symbols.
  - Let  $X$  be a variable symbol and  $P$  be a sentence.

Then  $(\forall X, P(X))$  is a sentence and  $(\exists X, P(X))$  is a sentence.

# Contd..

- **Universal quantification:**

- Often associated with English words “all”, “everyone”, “always”, etc.
- Syntax:  $\forall \langle \text{Variables} \rangle \langle \text{sentence} \rangle$
- E.g., Everyone at CAB is smart:

$$\forall x \text{ At}(x, \text{CAB}) \Rightarrow \text{Smart}(x)$$

(we can also read this as “if X is at CAB, then X is smart”)

- **$\forall x P(x)$  is true in a model M iff P(x) is true for all x in the model**

- Roughly speaking, equivalent to the conjunction of instantiations of P
- E.g.,:  $\text{At}(\text{Ram}, \text{CAB}) \Rightarrow \text{Smart}(\text{Ram}) \wedge \text{At}(\text{Hari}, \text{CAB}) \Rightarrow \text{Smart}(\text{Hari}) \wedge \dots$

# Contd..

- Typically,  $\Rightarrow$  is the main connective with  $\forall$ 
  - A universal quantifier is also equivalent to a set of implications over all objects
- **Common mistake:** using  $\wedge$  as the main connective with  $\forall$ :
  - $\forall x \text{ At}(x, \text{CAB}) \wedge \text{Smart}(x)$
  - Means “Everyone is at CAB and everyone is smart”
- You rarely use universal quantification to make blanket statements about every individual in the world (because such statement is hardly true)

e.g.,  $\forall x \text{ human}(x) \Rightarrow \text{mortal}(x)$

says, all humans are mortal

but,  $\forall x \text{ human}(x) \wedge \text{mortal}(x)$

say, everything is human and mortal

# Contd..

- **Existential quantification:**

- Often associated with English words “someone”, “sometimes”, etc.

- **Syntax:**  $\exists$ <variables> <sentence>

- Example: Someone at CAB is smart:

$$\exists x \text{ At}(x, \text{CAB}) \wedge \text{Smart}(x)$$

- **$\exists x P(x)$  is true in a model  $m$  iff  $P(x)$  is true for at least one  $x$  in the model**

- Roughly speaking, equivalent to the disjunction of instantiations of  $P$
- $\text{At}(\text{Ram}, \text{CAB}) \wedge \text{Smart}(\text{RAM}) \vee \text{At}(\text{Hari}, \text{CAB}) \wedge \text{Smart}(\text{Hari}) \vee \dots$

# Contd..

- Typically,  $\wedge$  is the main connective with  $\exists$
- Common mistake: using  $\Rightarrow$  as the main connective with  $\exists$ 
  - E.g.:  $\exists x \text{ At}(x, \text{CAB}) \Rightarrow \text{Smart}(x)$  is true even if there is anyone who is not at CAB!

e.g.,  $\exists x \text{ bird}(x) \wedge \neg \text{flies}(x)$

says, there is a bird that does not fly


but,  $\exists x \text{ bird}(x) \Rightarrow \neg \text{flies}(x)$

is also true for anything that is not a bird



# Quantifier Scope

- The part of a logical expression to which a quantifier is applied is called the scope of this quantifier.
- For example, suppose we want to say
  - “everyone who is alive loves someone”
  - $(\forall x) \text{ alive}(x) \rightarrow (\exists y) \text{ loves}(x, y)$
- Here’s how we scope the variables

$$(\forall x) \text{ alive}(x) \rightarrow (\exists y) \text{ loves}(x, y)$$


 Scope of x

 Scope of y

# Free vs. Bound Variables

- Definition

- An occurrence of a variable in a formula is bound iff the occurrence is in the scope of a quantifier employing the variable; otherwise it is free.

- Examples

- $\forall x.P(x, y)$
  - $x$  is bound
  - $y$  is free

# Nesting and mixing quantifiers

- Switching the order of multiple universal quantifiers does not change the meaning;

$$\forall X, \forall Y P(X,Y) \Leftrightarrow \forall Y, \forall X, P(X,Y)$$

- For example, “Brothers are siblings” can be written as

$$\forall x \forall y \text{ Brother}(x, y) \Rightarrow \text{Sibling}(x, y) .$$

- Consecutive quantifiers of the same type can be written as one quantifier with several variables.

$$\forall x, y \text{ Brother}(x, y) \Leftrightarrow \text{Sibling}(y, x) .$$

# Contd..

- Switching the order of multiple existential quantifiers does not change the meaning.

$$\exists X, \exists Y P(X,Y) \Leftrightarrow \exists Y, \exists X, P(X,Y)$$

# Contd..

- Switching the order of a universal quantifier and an existential quantifier does change meaning.

$\forall X, \exists Y P(X,Y)$  means "everyone loves someone"

$\exists Y, \forall X P(X,Y)$  means "someone is loved by everyone"

- The order of quantification is therefore very important.
- Use different variables with nested quantifiers to avoid confusion.

# Connections between All and Exists

- We can relate sentences involving  $\forall$  and  $\exists$  using **De Morgan's laws**:

1.  $(\forall x) \neg P(x) \leftrightarrow \neg(\exists x) P(x)$

2.  $\neg(\forall x) P(x) \leftrightarrow (\exists x) \neg P(x)$

3.  $(\forall x) P(x) \leftrightarrow \neg(\exists x) \neg P(x)$

4.  $(\exists x) P(x) \leftrightarrow \neg(\forall x) \neg P(x)$

- Examples

1. All dogs don't like cats  $\leftrightarrow$  No dogs like cats

2. Not all dogs dance  $\leftrightarrow$  There is a dog that doesn't dance

3. All dogs sleep  $\leftrightarrow$  There is no dog that doesn't sleep

4. There is a dog that talks  $\leftrightarrow$  Not all dogs can't talk

# Translating English to FOL

- Convert the following to the language of predicate logic.
  - Every apple is either green or yellow
  - No apple is blue
  - If an apple is green then its tasty
  - Every man likes a tasty apple
  - Some people like garlic
  - Fido is a dog and a good dog.
  - All basketball players are tall

# Translating English to FOL

Convert the following to the language of predicate logic.

- Every apple is either green or yellow

$$\forall X (apple(X) \Rightarrow green(X) \vee red(X))$$

- No apple is blue

$$\forall X (apple(x) \Rightarrow \neg blue(X))$$

- If an apple is green then its tasty

$$\forall X ((apple(X) \wedge green(X)) \Rightarrow tasty(X))$$

- Every man likes a tasty apple

$$\forall X \exists Y (man(X) \wedge tastyApple(Y) \Rightarrow likes(X, Y))$$

- Some people like garlic

$$\exists X (person(X) \Rightarrow likes(X, garlic))$$

- Fido is a dog and a good dog.

$$dog(fido) \wedge gooddog(fido)$$

- All basketball players are tall

$$\forall X (basketballPlayer(X) \Rightarrow tall(X))$$



# English to FOL conversion

- Snow is white
- Ram was a man
- Ram was a Nepali
- All Nepali are Asian
- Birendra was a ruler
- All Nepali were either loyal to Birendra or hated him.
- Everyone is loyal to someone.
- Gyanendra tried to assassinate Birendra.
- If it rains then sky will be cloudy.
- If you will not work hard, you will fail

# Contd..

- Ram likes sita.
- All indoor games are easy.
- Ram only likes cricket game
- All dogs are mammals.
- Roses are red.
- Ram is father of Hari.
- All the glitters is not gold
- Any person who is respected by every person is a king.
- God helps those who helps them selves.

# Contd..

- Ram likes to eat Peanut.
- Ram eats whatever John eats
- John eats eatable.
- Any body who eats eatable is a human.
- Peanut is an eatable.
- Ram likes all kinds of food.
- Apples are food.
- Chicken is food.

# Contd..

- Ram likes only easy courses.
- IT courses are hard.
- All courses in the computer department are easy.
- AI is an IT course.

# Translating English to FOL

All students are smart.

$\forall x ( \text{Student}(x) \Rightarrow \text{Smart}(x) )$

There exists a student.

$\exists x \text{ Student}(x).$

There exists a smart student.

$\exists x ( \text{Student}(x) \wedge \text{Smart}(x) )$

Every student loves some student.

$\forall x ( \text{Student}(x) \Rightarrow \exists y ( \text{Student}(y) \wedge \text{Loves}(x,y) ) )$

Every student loves some other student.

$\forall x ( \text{Student}(x) \Rightarrow \exists y ( \text{Student}(y) \wedge \neg (x = y) \wedge \text{Loves}(x,y) ) )$

There is a student who is loved by every other student.

$\exists x ( \text{Student}(x) \wedge \forall y ( \text{Student}(y) \wedge \neg (x = y) \Rightarrow \text{Loves}(y,x) ) )$

Bill is a student.

$\text{Student}(\text{Bill})$

Bill takes either Analysis or Geometry (but not both)

$\text{Takes}(\text{Bill}, \text{Analysis}) \Leftrightarrow \neg \text{Takes}(\text{Bill}, \text{Geometry})$

Bill takes Analysis or Geometry (or both).

$\text{Takes}(\text{Bill}, \text{Analysis}) \vee \text{Takes}(\text{Bill}, \text{Geometry})$

Bill takes Analysis and Geometry.

$\text{Takes}(\text{Bill}, \text{Analysis}) \wedge \text{Takes}(\text{Bill}, \text{Geometry})$

Bill does not take Analysis.

$\neg \text{Takes}(\text{Bill}, \text{Analysis}).$

# Contd..

No student loves Bill.

$\neg \exists x ( \text{Student}(x) \wedge \text{Loves}(x, \text{Bill}) )$

Bill has at least one sister.

$\exists x \text{ SisterOf}(x, \text{Bill})$

Bill has no sister.

$\neg \exists x \text{ SisterOf}(x, \text{Bill})$

Bill has at most one sister.

$\forall x, y ( \text{SisterOf}(x, \text{Bill}) \wedge \text{SisterOf}(y, \text{Bill}) \Rightarrow x = y )$

Bill has exactly one sister.

$\exists x ( \text{SisterOf}(x, \text{Bill}) \wedge \forall y ( \text{SisterOf}(y, \text{Bill}) \Rightarrow x = y ) )$

Bill has at least two sisters.

$\exists x, y ( \text{SisterOf}(x, \text{Bill}) \wedge \text{SisterOf}(y, \text{Bill}) \wedge \neg (x = y) )$

Every student takes at least one course.

$\forall x ( \text{Student}(x) \Rightarrow \exists y ( \text{Course}(y) \wedge \text{Takes}(x, y) ) )$

Only one student failed History.

$\exists x ( \text{Student}(x) \wedge \text{Failed}(x, \text{History}) \wedge \forall y ( \text{Student}(y) \wedge \text{Failed}(y, \text{History}) \Rightarrow x = y ) )$

No student failed Chemistry but at least one student failed History.

$\neg \exists x ( \text{Student}(x) \wedge \text{Failed}(x, \text{Chemistry}) ) \wedge \exists x ( \text{Student}(x) \wedge \text{Failed}(x, \text{History}) )$

Every student who takes Analysis also takes Geometry.

$\forall x ( \text{Student}(x) \wedge \text{Takes}(x, \text{Analysis}) \Rightarrow \text{Takes}(x, \text{Geometry}) )$

No student can fool all the other students.

$\neg \exists x ( \text{Student}(x) \wedge \forall y ( \text{Student}(y) \wedge \neg (x = y) \Rightarrow \text{Fools}(x, y) ) )$

# Translating English to FOL

- Every gardener likes the sun.

$(\forall x) \text{gardener}(x) \Rightarrow \text{likes}(x, \text{Sun})$

- Not Every gardener likes the sun.

$\sim((\forall x) \text{gardener}(x) \Rightarrow \text{likes}(x, \text{Sun}))$

- You can fool some of the people all of the time.

$(\exists x)(\forall t) \text{person}(x) \wedge \text{time}(t) \Rightarrow \text{can-be-fooled}(x, t)$

- You can fool all of the people some of the time.

$(\forall x)(\exists t) \text{person}(x) \wedge \text{time}(t) \Rightarrow \text{can-be-fooled}(x, t)$

(the time people are fooled may be different)

- You can fool all of the people at some time.

$(\exists t)(\forall x) \text{person}(x) \wedge \text{time}(t) \Rightarrow \text{can-be-fooled}(x, t)$

(all people are fooled at the same time)

- You can not fool all of the people all of the time.

$\sim((\forall x)(\forall t) \text{person}(x) \wedge \text{time}(t) \Rightarrow \text{can-be-fooled}(x, t))$

- Everyone is younger than his father

$(\forall x) \text{person}(x) \Rightarrow \text{younger}(x, \text{father}(x))$

Presented By: Tekendra Nath Yogi

# Translating English to FOL

- All purple mushrooms are poisonous.

$$(\forall x) (\text{mushroom}(x) \wedge \text{purple}(x)) \Rightarrow \text{poisonous}(x)$$

- No purple mushroom is poisonous.

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

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

- There are exactly two purple mushrooms.

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

$$(\forall z) (\text{mushroom}(z) \wedge \text{purple}(z)) \Rightarrow ((x=z) \vee (y=z))$$

- Clinton is not tall.

$$\sim\text{tall}(\text{Clinton})$$

- X is above Y if X is directly on top of Y or there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.

$$(\forall x)(\forall y) \text{above}(x,y) \Leftrightarrow (\text{on}(x,y) \vee (\exists z) (\text{on}(x,z) \wedge \text{above}(z,y)))$$



# Translating English to FOL

- Examples of English sentences converted to predicate logic
  - John loves Mary
    - Use the verb as the predicate and the nouns as arguments
      - loves(john, mary)
  - Mary is tall
    - Make a one argument predicate
      - tall(mary)
  - If the car is moving then wear the seat belt
    - Generally translate If X then Y into  $X \Rightarrow Y$ 
      - moving(car)  $\Rightarrow$  wear(seat\_belt)
  - If Paul is hungry then he eats fruit
    - Hungry(paul)  $\Rightarrow$  eats(paul, fruit)
  - All Students study
    - $\forall X \text{ student}(X) \Rightarrow \text{study}(X)$
  - There is something small and slimey on the floor
    - $\exists X ((\text{small}(X) \wedge \text{slimey}(X) \wedge \text{onfloor}(X))$

# Translating English to FOL

$\text{Cat}(\text{herschel}) \wedge \text{Cat}(\text{orfy})$

*Herschel is a cat and Orfy is a cat.*

$\text{Dog}(\text{heidi}) \wedge \text{Dog}(\text{willy})$

*Heidi and Willy are dogs.*

$\text{Loves}(\text{heidi}, \text{gary})$

*Heidi loves Gary.*

$\text{Fed}(\text{gary}, \text{heidi}, 2:00)$

*Gary fed Heidi at 2 pm.*

$\neg(\text{Dog}(\text{orfy}) \wedge \text{Dog}(\text{willy}))$

*Not both Orfy and Willy are dogs.*

$\neg(\text{Cat}(\text{willy}) \vee \text{Cat}(\text{heidi}))$

*Neither Willy nor Heidi is a cat.*

$\text{Dog}(\text{willy}) \rightarrow \neg \text{Person}(\text{willy})$

*If Willy is a dog, then he isn't a person.*

$\exists x (\text{Pet}(x) \wedge \text{Dog}(x))$

*Some pets are dogs.*

$\exists x (\text{Pet}(x) \wedge \text{Cat}(x))$

*Some cats are pets.*

$\forall x (\text{Pet}(x) \rightarrow (\text{Dog}(x) \vee \text{Cat}(x)))$

*Every pet is either a dog or a cat.*

$\forall x (\text{Dog}(x) \rightarrow \neg \text{Cat}(x))$

*No dog is a cat.*

$\neg \exists x (\text{Dog}(x) \wedge \text{Cat}(x))$

*No dog is a cat.*

$\forall x (\text{Cat}(x) \rightarrow \text{Larger}(\text{willy}, x))$

*Willy is larger than every cat.*

$\neg \exists x \text{Fed}(\text{elly}, \text{willy}, x)$

*Elly has never fed Willy.*

# FOPL: Semantic

- An interpretation is required to give semantics to first-order logic. The interpretation is a non-empty “domain of discourse” (set of objects). The truth of any formula depends on the interpretation. The interpretation provides, for each:
  - **constant symbol** an object in the domain
  - **function symbols** a function from domain tuples to the domain
  - **predicate symbol** a relation over the domain (a set of tuples)
- Then we define:
  - **universal quantifier**  $\forall xP(x)$  is True iff  $P(a)$  is True for all assignments of domain elements  $a$  to  $x$
  - **existential quantifier**  $\exists xP(x)$  is True iff  $P(a)$  is True for at least one assignment of domain element  $a$  to  $x$

# Interpretations & Models in FOPL

- **Definition:** An *interpretation* is a mapping which assigns
  - objects in domain to constants in the language
  - functional relationships in domain to function symbols
  - relations to predicate symbols
  - usual logical relationships to connectives and quantifiers:  $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \forall, \exists$
- **Definition: Models**
  - An interpretation  $M$  is a model for a set of sentences  $S$ , if every sentence in  $S$  is true with respect to  $M$ .

# Interpretations & Models in FOPL

- Example:  $s = \forall x N(x) \Rightarrow L(x, f(x))$

where  $N, L$  are predicate symbols, and  $f$  a function symbol

## – Interpretation 1

- domain = positive integers
- $N(x)$  = “ $x$  is a natural number”
- $L(x,y)$  = “ $x$  is less than  $y$ ”
- $f(x)$  = “predecessor of  $x$ ” (i.e.,  $x-1$ )
- then  $s$  says: “any natural number is a less than its predecessor” (of course this is false, so this interpretation is not a model for  $s$ )

## – Interpretation 2

- domain = all people
- $N(x)$  = “ $x$  is a person”
- $L(x,y)$  = “ $x$  likes  $y$ ”
- $f(x)$  = “mother of  $x$ ”
- then  $s$  says: “everyone likes his/her mother”



# FOPL Inference

Presented By : Tekendra Nath Yogi

[Tekendranath@gmail.com](mailto:Tekendranath@gmail.com)

College Of Applied Business And Technology



# FOPL Inference

- Three major families of first-order inference algorithms:
  - **Forward chaining**
  - **Backward chaining**
  - **Resolution**

# Contd..

- **Idea:** first-order inference can be done by converting the knowledge base to propositional logic and using propositional inference.
- Conversion can be done by using the following inference rules for quantifiers:
  - Universal instantiation
  - Existential instantiation
- applied to sentences with quantifiers to obtain sentences without quantifiers.



# Contd...

- Universal instantiation (UI):
  - **UI Rule:** Substitute ground term (term without variables) for the variables.
  - a universally quantified sentence can be replaced by the set of all possible instantiations.
  - After UI we discard the universally quantified sentence.

# Contd..

- **For example:** suppose our knowledge base contains just the sentences
  - $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$  (all greedy kings are evil)
  - $\text{King}(\text{John})$
  - $\text{Greedy}(\text{John})$
  - $\text{Brother}(\text{Richard}, \text{John})$  .
- Then we apply UI to the first sentence using all possible ground-term substitutions from the vocabulary of the knowledge base (in this case,  $\{x/\text{John}\}$  and  $\{x/\text{Richard}\}$ ).
- We obtain
  - $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$
  - $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$  ,
- and we discard the universally quantified sentence.

# Contd..

- **Existential instantiation(EI):**
  - **EI rule:** the variable is replaced by a single new constant symbol does not appear elsewhere in the knowledge base.
  - Basically, the existential sentence says there is some object satisfying a condition, and applying the existential instantiation rule just gives a name to that object.
  - So, Existential Instantiation can be applied once, and then the existentially quantified sentence can be discarded.
  - **For example**, we no longer need  $\exists x \text{ Kill}(x, \text{Ram})$  once we have added the sentence **Kill (Hari , Ram)**.

# Contd..

For example, from the sentence

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

we can infer the sentence

$$\text{Crown}(C_1) \wedge \text{OnHead}(C_1, \text{John})$$

as long as  $C_1$  does not appear elsewhere in the knowledge base.

- After the universal instantiation and existential instantiation, the knowledge base is propositional.
- Therefore, we can apply any of the complete propositional inference algorithms

# Simple Proof Example1

- Suppose we have the following sentences in the KB:
  - Anything that barks is a dog.
  - Fido barks.
  - prove that Fido is a dog.

# Contd..

- Step 1: State the facts you know in FOL
  - We know anything that barks is a dog. State this fact in FOL:
    - a.  $\forall x \text{Barks}(x) \Rightarrow \text{Dog}(x)$ 
      - This says “If x barks then x is a dog”.
  - We know that Fido barks. State this fact also in FOL:
    - b.  $\text{Barks}(\text{Fido})$

# Contd..

- Step 2: Remove all quantifiers
  - Apply the Universal Instantiation inference rule to remove the universal quantifier ( $\forall$ ) in sentence a
  - The result is:  
c.  $\text{Barks}(\text{Fido}) \Rightarrow \text{Dog}(\text{Fido})$

# Contd..

- Now KB in propositional form is:

Barks(Fido)  
Barks(Fido)  $\Rightarrow$  Dog(Fido)

- Step 3: See what inference rules can be applied.
  - Think about what we want to do: Eliminate the implication, leaving the sentence “Dog(Fido)”
  - How can we do this?
  - Use Modus Ponens

D. Dog(Fido)



# Inference example2

- Consider the following:
  - "All bill's dogs are brown. Bill owns fred. Fred is a dog".
  - Can we infer "fred is brown"?
  
- **TRANSLATE INTO FIRST ORDER LOGIC:**
  - "All bill's dogs are brown" -  
$$\forall X \text{ DOG}(X) \wedge \text{OWNS}(\text{bill}, X) \Rightarrow \text{BROWN}(X)$$
  - "Bill owns fred" -  $\text{OWNS}(\text{bill}, \text{fred})$
  - "Fred is a dog" -  $\text{DOG}(\text{fred})$
  
- **APPLY INFERENCING RULES:**
  - AND-INTRODUCTION -  $\text{DOG}(\text{fred}) \wedge \text{OWNS}(\text{bill}, \text{fred})$
  - UNIVERSAL ELIMINATION -  
$$\text{DOG}(\text{fred}) \wedge \text{OWNS}(\text{bill}, \text{fred}) \Rightarrow \text{BROWN}(\text{fred})$$
  - MODUS PONENS -  $\text{BROWN}(\text{fred})$

# Generalized Modus ponens

- Combination of AND- introduction, Universal elimination, and modus ponens into one step.
- If there is some substitution  $\theta$  {e.g.,  $x/\text{John}$ } that makes each of the conjuncts of the premise of the implication identical to sentences already in the knowledge base, then we can assert the conclusion of the implication, after applying  $\theta$ .
  - **KB:**
    - $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
    - $\text{King}(\text{John})$
    - $\text{Greedy}(\text{John})$
  - **Query:**
    - $\text{Evil}(\text{John})$
  - For this query, we can infer  $\text{Evil}(\text{john})$  from the given KB with the substitution  $\theta = \{x/\text{John}\}$

# Unification

- Unification is the process of finding substitutions that make different logical expressions look identical.
- takes two sentences and returns a list of substitutions(unifier) to make two sentences match, or failure if no match possible.
- i.e.,  $\text{UNIFY}(p, q) = \theta$  , are matched where  $\theta$  is the list of substitutions in  $p$  and  $q$

# Contd..

- **Unification rules:**

1. Function symbols and predicate symbols must have identical names and number of arguments.
2. Constant symbols unify with only identical constant symbols.
3. Variables unify with other variable symbols, constant symbols or function symbols
4. Variable symbols may not be unified with other terms in which the variable itself occurs.
  - **For example:**  $x$  can not unify with  $G(x)$  since this will lead to  $G(G(G(\dots G(x))))$

# Contd..

- Example: unification

p	q	$\theta$
Knows(John,x)	Knows(John,Jane)	$\{x/Jane\}$
Knows(John,x)	Knows(y,OJ)	$\{x/OJ, y/John\}$
Knows(John,x)	Knows(y,Mother(y))	$\{y/John, x/Mother(John)\}$
Knows(John,x)	Knows(x,OJ)	$\{fail\}$

- Last unification is failed due to overlap of variables
- i.e., x can not take the values of John and OJ at the same time.
- We can avoid this problem(name clashes ) by renaming ( standardizing apart)
- E.g.,  $Unify\{Knows(John,x) \quad Knows(z,OJ)\} = \{x/OJ, z/John\}$

# RESOLUTION

- **Conjunctive normal form for first-order logic :**
  - As in the propositional case, first-order resolution requires that sentences be in conjunctive normal form (CNF)—that is, a **conjunction of clauses**, where each **clause is a disjunction of literals**.
  - Literals can contain variables, which are assumed to be universally quantified.
  - **For example:**
    - $[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$

# Contd..

- **Conversion to CNF**

- The procedure for conversion to CNF is similar to the propositional case.
- We illustrate the procedure by translating the sentence “Everyone who loves all animals is loved by someone,” or
  - $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{ Loves}(y, x)] .$

## Step1: Eliminate Bi-Implications and implications

$$\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Loves}(x, y)] \vee [\exists y \text{ Loves}(y, x)] .$$

# Contd..

- Step2: Move  $\neg$  inwards

- In addition to the usual rules for negated connectives, we need rules for negated quantifiers. Thus, we have

- $\neg \forall x p$  becomes  $\exists x \neg p$
- $\neg \exists x p$  becomes  $\forall x \neg p$ .

- Our sentence becomes:

- $\forall x [\exists y \neg(\neg \text{Animal}(y) \vee \text{Loves}(x, y))] \vee [\exists y \text{Loves}(y, x)]$ .
- $\forall x [\exists y \neg\neg \text{Animal}(y) \wedge \neg \text{Loves}(x, y)] \vee [\exists y \text{Loves}(y, x)]$ .



# Contd..

- Step3: Eliminate double negation ( $\neg\neg$ )
  - $\forall x [\exists y \text{ Animal}(y) \wedge \neg\neg\text{Loves}(x, y)] \vee [\exists y \text{ Loves}(y, x)] .$

# Contd..

- **Step4: Standardize variables (Rename)**
  - Rename bound variables so that each only occurs once
  - For sentences like  $(\exists x P(x)) \vee (\exists x Q(x))$  which use the same variable name twice, change the name of one of the variables.
    - Thus, we have
    - $\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Loves}(x, y)] \vee [\exists z \text{ Loves}(z, x)]$  .

# Contd..

- Step 5 : Move quantifiers to the left

- $(\forall x P(x)) \vee Q$

- Can be written as:  $\forall x (P(x) \vee Q)$  and

- $(\exists x P(x)) \vee (\exists y Q(y))$

- Can be written as:  $\exists x \exists y (P(x) \vee Q(y))$

- Thus, we have

- $\forall x \exists y \exists z [ \text{Animal}(y) \wedge \neg \text{Loves}(x, y) ] \vee \text{Loves}(z, x) ]$ .

# Contd..

- **Step 6: Skolemize to eliminate existential quantifiers**
  - Skolemization is the process of removing existential quantifiers by elimination.
  - If  $\exists$  is not in the scope of  $\forall$  then eliminate  $\exists$  and replace existentially quantified variable by a constant not in the knowledge base.
  - E.g., a translate  $\exists x P(x)$  into  $P(A)$ , where  $A$  is a new constant.
  - If  $\exists$  is in the scope of  $\forall$  then eliminate  $\exists$  and replace existentially quantified variable by a function with argument universally quantified variables in whose scope the existential quantifier appears.
    - $\forall x [\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$  .
    - Here  $F$  and  $G$  are Skolem functions.

# Contd..

- **Step7: Drop universal quantifiers**
  - At this point, all remaining variables must be universally quantified. Moreover, the sentence is equivalent to one in which all the universal quantifiers have been moved to the left. We can therefore drop the universal quantifiers:
  - $[\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x) .$

# Contd..

- Step8: Distribute  $\vee$  over  $\wedge$

- $[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$  .
- This step may also require flattening out nested conjunctions and disjunctions.
- The sentence is now in CNF and consists of two clauses.
  - $[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)]$
  - $[\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$

# Contd..

- **CNF Conversion summarized algorithm:**

- Above descriptive steps of CNF conversion can be summarized in the following steps

1. Eliminate implications and bi-implications as in propositional case

2. Move negations inward using De Morgan's laws

plus rewriting  $\neg \forall x P$  as  $\exists x \neg P$  and  $\neg \exists x P$  as  $\forall x \neg P$

3. Eliminate double negations

4. Rename bound variables if necessary so each only occurs once

e.g.  $\forall x P(x) \vee \exists x Q(x)$  becomes  $\forall x P(x) \vee \exists y Q(y)$

5. Use equivalences to move quantifiers to the left

e.g.  $\forall x P(x) \wedge Q$  becomes  $\forall x (P(x) \wedge Q)$  where  $x$  is not in  $Q$

e.g.  $\forall x P(x) \wedge \exists y Q(y)$  becomes  $\forall x \exists y (P(x) \wedge Q(y))$

6. Skolemise (replace each existentially quantified variable by a **new** term)

$\exists x P(x)$  becomes  $P(a_0)$  using a Skolem constant  $a_0$  since  $\exists x$  occurs at the outermost level

$\forall x \exists y P(x, y)$  becomes  $P(x, f_0(x))$  using a Skolem function  $f_0$  since  $\exists y$  occurs within  $\forall x$

7. The formula now has only universal quantifiers and all are at the left of the formula: drop them

8. Use distribution laws to get CNF and then clausal form

# The resolution inference rule:

- Two clauses, which are assumed to be standardized apart so that they share no variables, can be resolved if they contain complementary literals.
- Propositional literals are complementary if one is the negation of the other; first-order literals are complementary if one unifies with the negation of the other. Thus, we have

$$\frac{\ell_1 \vee \dots \vee \ell_k, \quad m_1 \vee \dots \vee m_n}{\text{SUBST}(\theta, \ell_1 \vee \dots \vee \ell_{i-1} \vee \ell_{i+1} \vee \dots \vee \ell_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n)}$$

where  $\text{UNIFY}(\ell_i, \neg m_j) = \theta$



# Contd..

- For example, we can resolve the two clauses

$$[Animal(F(x)) \vee Loves(G(x), x)] \quad \text{and} \quad [\neg Loves(u, v) \vee \neg Kills(u, v)]$$

by eliminating the complementary literals  $Loves(G(x), x)$  and  $\neg Loves(u, v)$ , with unifier  $\theta = \{u/G(x), v/x\}$ , to produce the **resolvent** clause

$$[Animal(F(x)) \vee \neg Kills(G(x), x)] .$$

# Resolution Algorithm

- **Algorithm:**
  - Convert KB into first order logic expressions.
  - Convert knowledge base (FOPL logic expressions) into CNF
  - convert the negation of query into CNF and then add them into KB.
  - Repeatedly apply resolution to clauses or copies of clauses(a copy of a clause is the clause with all variables renamed) until either the empty clause is derived or no more clauses can be derived.
    - If the empty clause is derived , answer = Yes ( query follows from knowledge base).
    - Otherwise answer = No ( query does not follow from knowledge base)

# Example of resolution refutation

- **Example:** Consider the following statements:
  - Everyone who loves all animal is loved by some one.
  - Jack is loves all animal
  - Query: Jack is loved by someone.

# Contd..

- KB in FOPL:
  - $\forall x [\forall y \text{Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{Loves}(y, x)] .$
  - $\forall y \text{Animal}(y) \Rightarrow \text{Loves}(\text{jack}, y)$
- Query in FOPL:
  - $\exists y \text{Animal}(y) \Rightarrow \text{Loves}(y, \text{jack})$
- Negatón of query:
  - $\forall y \neg \text{Animal}(y) \Rightarrow \neg \text{Loves}(y, \text{jack})$

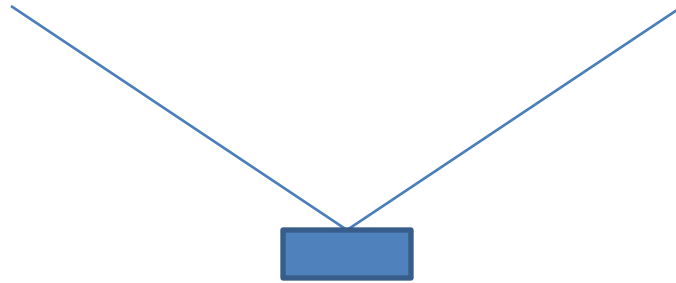
# Contd..

- KB and Query in CNF:
  - $[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)]$
  - $[\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$
  - $\neg \text{Animal}(y) \vee \text{Loves}(\text{jack}, y)$
  - $\text{Animal}(y) \vee \neg \text{Loves}(y, \text{jack})$

# Contd..

$\neg \text{Animal}(y) \vee \text{Loves}(\text{jack}, y)$

$\text{Animal}(y) \vee \neg \text{Loves}(y, \text{jack})$



Hence jack is loved by someone

# Example of resolution refutation

**KB:**

1. Anyone passing his history exams and winning the lottery is happy.
2. Anyone who studies or is lucky can pass all his exams.
3. John did not study but John is lucky.
4. Anyone who is lucky wins the lottery.

**query:** “Is John happy?”.

Use the resolution refutation algorithm to answer the given query.

# Contd..

(a) Translate the following four English sentences to first order logic (FOL).

1. Anyone passing his history exams and winning the lottery is happy.
2. Anyone who studies or is lucky can pass all his exams.
3. John did not study but John is lucky.
4. Anyone who is lucky wins the lottery.

(b) Convert them to conjunctive normal form (CNF).

(c) Answer the query “Is John happy?”. Use the resolution refutation algorithm.



## Contd...

**(a) Translate the following four English sentences to first order logic (FOL).**

1. Anyone passing his history exams and winning the lottery is happy.

$$\forall x \text{ Pass}(x, \text{HistoryExam}) \wedge \text{Win}(x, \text{Lottery}) \Rightarrow \text{Happy}(x)$$

2. Anyone who studies or is lucky can pass all his exams.

3. John did not study but John is lucky.

4. Anyone who is lucky wins the lottery.

# Contd..

1. Anyone passing his history exams and winning the lottery is happy.

$$\forall x \text{ Pass}(x, \text{HistoryExam}) \wedge \text{Win}(x, \text{Lottery}) \Rightarrow \text{Happy}(x)$$

2. Anyone who studies or is lucky can pass all his exams.

$$\forall x \forall y \text{ Study}(x) \vee \text{Lucky}(x) \Rightarrow \text{Pass}(x, y)$$

3. John did not study but John is lucky.

4. Anyone who is lucky wins the lottery.

# Contd..

1. Anyone passing his history exams and winning the lottery is happy.

$$\forall x \text{ Pass}(x, \text{HistoryExam}) \wedge \text{Win}(x, \text{Lottery}) \Rightarrow \text{Happy}(x)$$

2. Anyone who studies or is lucky can pass all his exams.

$$\forall x \forall y \text{ Study}(x) \vee \text{Lucky}(x) \Rightarrow \text{Pass}(x, y)$$

3. John did not study but John is lucky.

$$\neg \text{Study}(\text{John}) \wedge \text{Lucky}(\text{John})$$

4. Anyone who is lucky wins the lottery.

# Contd..

1. Anyone passing his history exams and winning the lottery is happy.

$$\forall x \text{ Pass}(x, \text{HistoryExam}) \wedge \text{Win}(x, \text{Lottery}) \Rightarrow \text{Happy}(x)$$

2. Anyone who studies or is lucky can pass all his exams.

$$\forall x \forall y \text{ Study}(x) \vee \text{Lucky}(x) \Rightarrow \text{Pass}(x, y)$$

3. John did not study but John is lucky.

$$\neg \text{Study}(\text{John}) \wedge \text{Lucky}(\text{John})$$

4. Anyone who is lucky wins the lottery.

$$\forall x \text{ Lucky}(x) \Rightarrow \text{Win}(x, \text{Lottery})$$

## (b) Convert them to conjunctive normal form (CNF).

$$1 \quad \forall x \text{ Pass}(x, \text{HistoryExam}) \wedge \text{Win}(x, \text{Lottery}) \Rightarrow \text{Happy}(x)$$

$$2 \quad \forall x \forall y \text{ Study}(x) \vee \text{Lucky}(x) \Rightarrow \text{Pass}(x, y)$$

$$3 \quad \neg \text{Study}(\text{John}) \wedge \text{Lucky}(\text{John})$$

$$4 \quad \forall x \text{ Lucky}(x) \Rightarrow \text{Win}(x, \text{Lottery})$$

First: Implication elimination

# Contd..

$$1 \quad \neg [Pass(x, HistoryExam) \wedge Win(x, Lottery)] \vee Happy(x)$$

$$2 \quad \neg [Study(x) \vee Lucky(x)] \vee Pass(x, y)$$

$$3 \quad \neg Study(John) \wedge Lucky(John)$$

$$4 \quad \neg Lucky(x) \vee Win(x, Lottery)$$

Then: drop the "for all" quantifiers

# Contd..

$$1 \quad [\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery)] \vee Happy(x)$$

$$2 \quad [\neg Study(x) \wedge \neg Lucky(x)] \vee Pass(x, y)$$

$$3 \quad \neg Study(John) \wedge Lucky(John)$$

$$4 \quad \neg Lucky(x) \vee Win(x, Lottery)$$

Then: move the negation inwards

# Contd...

$$1 \quad \neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$$

$$2 \quad [\neg Study(x) \vee Pass(x, y)] \wedge [\neg Lucky(x) \vee Pass(x, y)]$$

$$3 \quad \neg Study(John) \wedge Lucky(John)$$

$$4 \quad \neg Lucky(x) \vee Win(x, Lottery)$$

Then: distribute the "ors"



# Contd..

- 1       $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$
- 2a      $\neg \text{Study}(x) \vee \text{Pass}(x, y)$
- 2b      $\neg \text{Lucky}(x) \vee \text{Pass}(x, y)$
- 3a      $\neg \text{Study}(\text{John})$
- 3b      $\text{Lucky}(\text{John})$
- 4       $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$

Then: separate the "and" sentences into individual sentences

## (c) Query and resolution refutation

- 1  $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$
- 2a  $\neg \text{Study}(x) \vee \text{Pass}(x, y)$
- 2b  $\neg \text{Lucky}(x) \vee \text{Pass}(x, y)$
- 3a  $\neg \text{Study}(\text{John})$
- 3b  $\text{Lucky}(\text{John})$
- 4  $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$

Knowledge Base (KB)

# Contd..

**Note: Construct the resolution tree for the clauses given below as in propositional logic but use the predicate resolution rule instead of resolution rule of propositional logic.**

1       $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$

2a      $\neg \text{Study}(x) \vee \text{Pass}(x, y)$

2b      $\neg \text{Lucky}(x) \vee \text{Pass}(x, y)$

3a      $\neg \text{Study}(\text{John})$

3b      $\text{Lucky}(\text{John})$

4       $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$

**Q**       $\neg \text{Happy}(\text{John})$

The negation of our query

# Contd..

1  $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$

2a  $\neg \text{Study}(x) \vee \text{Pass}(x, y)$

2b  $\neg \text{Lucky}(x) \vee \text{Pass}(x, y)$

{x/John}

3a  $\neg \text{Study}(\text{John})$

3b  $\text{Lucky}(\text{John})$

4  $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$

Q  $\neg \text{Happy}(\text{John})$

# Contd..

1  $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$

2a  $\neg \text{Study}(x) \vee \text{Pass}(x, y)$

2b  $\neg \text{Lucky}(x) \vee \text{Pass}(x, y)$

{x/John}

3a  $\neg \text{Study}(\text{John})$

3b  $\text{Lucky}(\text{John})$

4  $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$

$\neg \text{Pass}(\text{John}, \text{HistoryExam}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$

5

Q  $\neg \text{Happy}(\text{John})$

# Contd..

1  $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$

2a  $\neg \text{Study}(x) \vee \text{Pass}(x, y)$

2b  $\neg \text{Lucky}(x) \vee \text{Pass}(x, y)$

3a  $\neg \text{Study}(\text{John})$

{x/John; y/HistoryExam}

3b  $\text{Lucky}(\text{John})$

4  $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$

$\neg \text{Pass}(\text{John}, \text{HistoryExam}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$

5

Q  $\neg \text{Happy}(\text{John})$

# Contd..

1  $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$

2a  $\neg \text{Study}(x) \vee \text{Pass}(x, y)$

2b  $\neg \text{Lucky}(x) \vee \text{Pass}(x, y)$

3a  $\neg \text{Study}(\text{John})$

$\{x/\text{John}; y/\text{HistoryExam}\}$

3b  $\text{Lucky}(\text{John})$

4  $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$

$\neg \text{Pass}(\text{John}, \text{HistoryExam}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$

5

$\neg \text{Lucky}(\text{John}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$

6

# Contd..

1  $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$

2a  $\neg \text{Study}(x) \vee \text{Pass}(x, y)$

2b  $\neg \text{Lucky}(x) \vee \text{Pass}(x, y)$

3a  $\neg \text{Study}(\text{John})$

3b  $\text{Lucky}(\text{John})$

4  $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$   $\{x/\text{John}\}$

$\neg \text{Pass}(\text{John}, \text{HistoryExam}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$  5

$\neg \text{Lucky}(\text{John}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$  6



# Contd..

1  $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$

2a  $\neg \text{Study}(x) \vee \text{Pass}(x, y)$

$\neg \text{Lucky}(\text{John})$

7

3a  $\neg \text{Study}(\text{John})$

3b  $\text{Lucky}(\text{John})$

4  $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$   $\{x/\text{John}\}$

$\neg \text{Pass}(\text{John}, \text{HistoryExam}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$

5

$\neg \text{Lucky}(\text{John}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$

6

# Contd..

1  $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$

2a  $\neg \text{Study}(x) \vee \text{Pass}(x, y)$

$\neg \text{Lucky}(\text{John})$

7

3a  $\neg \text{Study}(\text{John})$

3b  $\text{Lucky}(\text{John})$

4  $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$

$\neg \text{Pass}(\text{John}, \text{HistoryExam}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$

5

$\neg \text{Lucky}(\text{John}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$

6

# Contd..

1  $\neg \text{Pass}(x, \text{HistoryExam}) \vee \neg \text{Win}(x, \text{Lottery}) \vee \text{Happy}(x)$

2a  $\neg \text{Study}(x) \vee \text{Pass}(x, y)$

$\neg \text{Lucky}(\text{John})$

7

3a  $\neg \text{Study}(\text{John})$

3b  $\text{Lucky}(\text{John})$

$\emptyset$

4  $\neg \text{Lucky}(x) \vee \text{Win}(x, \text{Lottery})$

$\neg \text{Pass}(\text{John}, \text{HistoryExam}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$

5

$\neg \text{Lucky}(\text{John}) \vee \neg \text{Win}(\text{John}, \text{Lottery})$

6

# Resolution Example2

- In English, the problem is as follows:
  - Everyone who loves all animals is loved by someone.
  - Anyone who kills an animal is loved by no one.
  - Jack loves all animals.
  - Either Jack or Curiosity killed the cat, who is named Tuna.
  - Did Curiosity kill the cat?

# Contd..

- First, we express the original sentences, some background knowledge, and the negated goal G in first-order logic:

- A.  $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{ Loves}(y, x)]$
- B.  $\forall x [\exists z \text{ Animal}(z) \wedge \text{Kills}(x, z)] \Rightarrow [\forall y \neg \text{Loves}(y, x)]$
- C.  $\forall x \text{ Animal}(x) \Rightarrow \text{Loves}(\text{Jack}, x)$
- D.  $\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})$
- E.  $\text{Cat}(\text{Tuna})$
- F.  $\forall x \text{ Cat}(x) \Rightarrow \text{Animal}(x)$
- ¬G.  $\neg \text{Kills}(\text{Curiosity}, \text{Tuna})$

# Contd..

- Now we apply the CNF conversion procedure to convert each sentence to CNF.

A1.  $Animal(F(x)) \vee Loves(G(x), x)$

A2.  $\neg Loves(x, F(x)) \vee Loves(G(x), x)$

B.  $\neg Loves(y, x) \vee \neg Animal(z) \vee \neg Kills(x, z)$

C.  $\neg Animal(x) \vee Loves(Jack, x)$

D.  $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

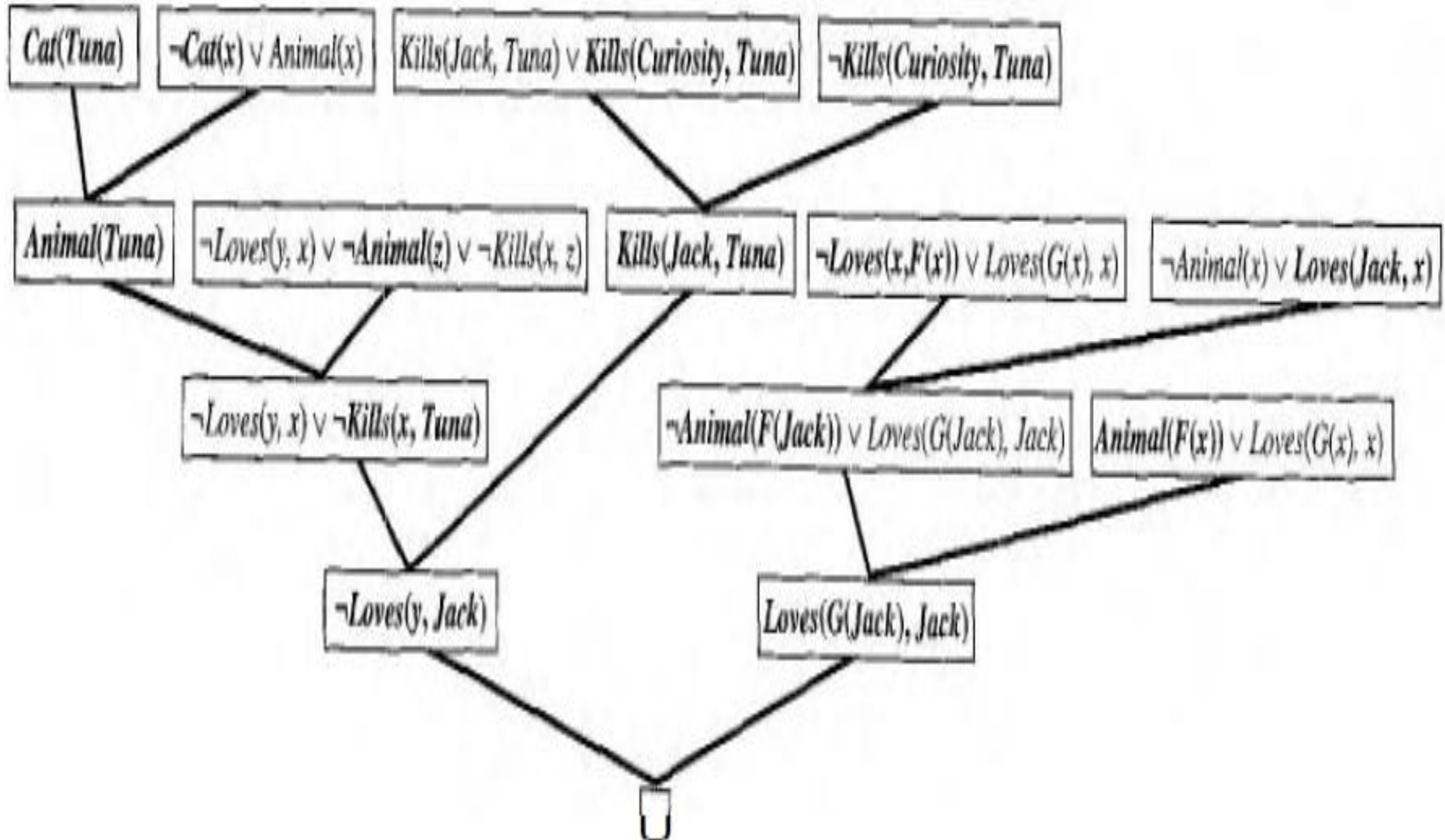
E.  $Cat(Tuna)$

F.  $\neg Cat(x) \vee Animal(x)$

$\neg$ G.  $\neg Kills(Curiosity, Tuna)$

# Contd..

- The resolution proof that Curiosity killed the cat is given in Figure below:





# Semantic Network

Presented By : Tekendra Nath Yogi

[Tekendranath@gmail.com](mailto:Tekendranath@gmail.com)

College Of Applied Business And Technology





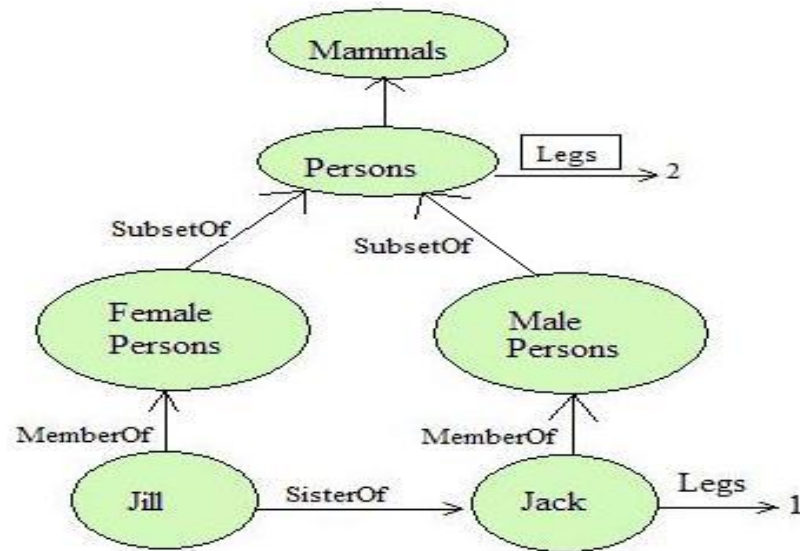
# Semantic Network

- A **semantic net** (or semantic network) is a knowledge representation technique used for propositional information. So it is also called a **propositional net**.
- Mathematically a semantic net can be defined as a labeled directed graph.
- consist of:
  - nodes,
  - links (edges) and
  - link labels.

# Semantic Network

- **Nodes:**
  - In the semantic network diagram, nodes appear as circles or ellipses or rectangles to represent objects such as physical objects, concepts or situations.
- **Links:**
  - appear as arrows to express the relationships between objects, and
- **link labels:**
  - specify particular relations. Relationships provide the basic structure for organizing knowledge.
  - The objects and relations involved need not be so concrete.
- As nodes are associated with other nodes semantic nets are also referred to as associative nets.

# Semantic Network



- In the above figure all the objects are within ovals and connected using labelled arcs.
- Note that there is a link between Jill and FemalePersons with label MemberOf. Similarly there is a MemberOf link between Jack and MalePersons and SisterOf link between Jill and Jack.
- The MemberOf link between Jill and FemalePersons indicates that Jill belongs to the category of female persons.

# Semantic Network

- **Inheritance Reasoning**

- Unless there is a specific evidence to the contrary, it is assumed that all members of a class (category) will inherit all the properties of their super classes.
- So semantic network allows us to perform inheritance reasoning.
  - **For example:** Jill inherits the property of having two legs as she belongs to the category of FemalePersons which in turn belongs to the category of Persons which has a boxed Legs link with value 2.
- Semantic nets allows multiple inheritance. So an object can belong to more than one category and a category can be a subset of more than one another category.

# Semantic Network

- Inverse Links

- Semantic network allows a common form of inference known as inverse links.

- For example: we can have a *HasSister* link which is the inverse of *SisterOf* link.
    - The inverse links make the job of inference algorithms much easier to answer queries such as who the sister of *Jack* is.
    - On discovering that *HasSister* is the inverse of *SisterOf* the inference algorithm can follow that link *HasSister* from *Jack* to *Jill* and answer the query.

# Semantic Network

- **Disadvantage Of Semantic Nets**
  - One of the drawbacks of semantic network is that the links between the objects represent only binary relations.
    - For example, the sentence Run(Kirtipur Express, Kirtipur, Ratnapark, Today) cannot be asserted directly.
  - There is no standard definition of link names.

# Semantic Network

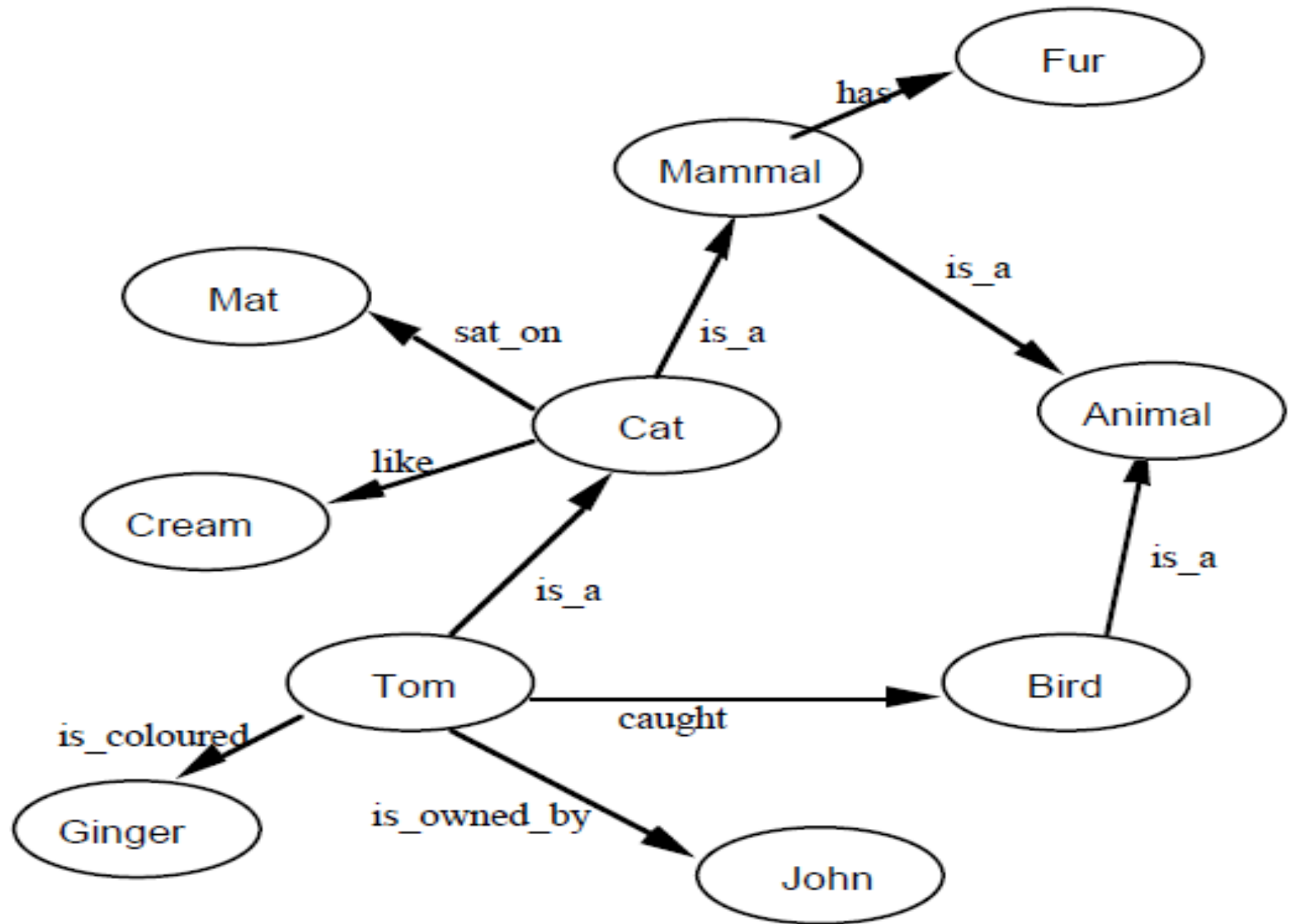
- Advantages Of Semantic Nets
  - Semantic nets have the ability to represent default values for categories.
    - In the above figure Jack has one leg while he is a person and all persons have two legs. So persons have two legs has only default status which can be overridden by a specific value.
  - They convey some meaning in a transparent manner.
  - Semantic nets are simple and easy to understand.

# Example1: semantic network

- Represent the following fact in semantic network
  - Tom is a cat.
  - Tom caught a bird.
  - Tom is owned by John.
  - Tom is ginger in color.
  - Cats like cream.
  - The cat sat on the mat.
  - A cat is a mammal.
  - A bird is an animal.
  - All mammals are animals.
  - Mammals have fur.



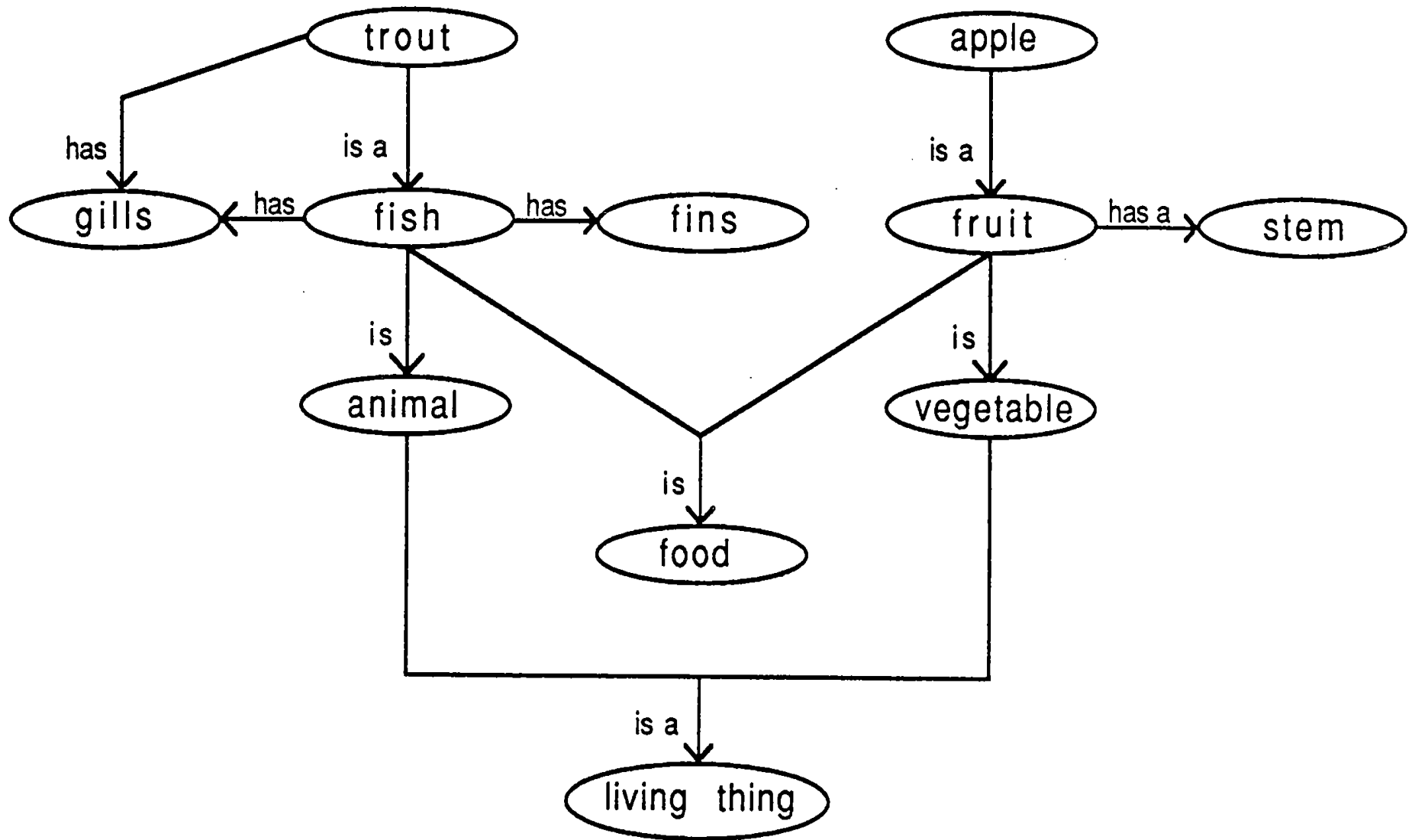
# Contd..



## Example2: semantic network

- Represent the following fact in semantic network
  - (1) A trout is a fish.
  - (2) A fish has gills.
  - (3) A fish has fins.
  - (4) Fish is food.
  - (5) Fish is animal.
  - (6) An apple is a fruit.
  - (7) Fruit has a stem.
  - (8) Fruit is food.
  - (9) Fruit is vegetable.
  - (10) An animal is a living thing.
  - (11) A vegetable is a living thing.

# Contd..



# Thank You !

