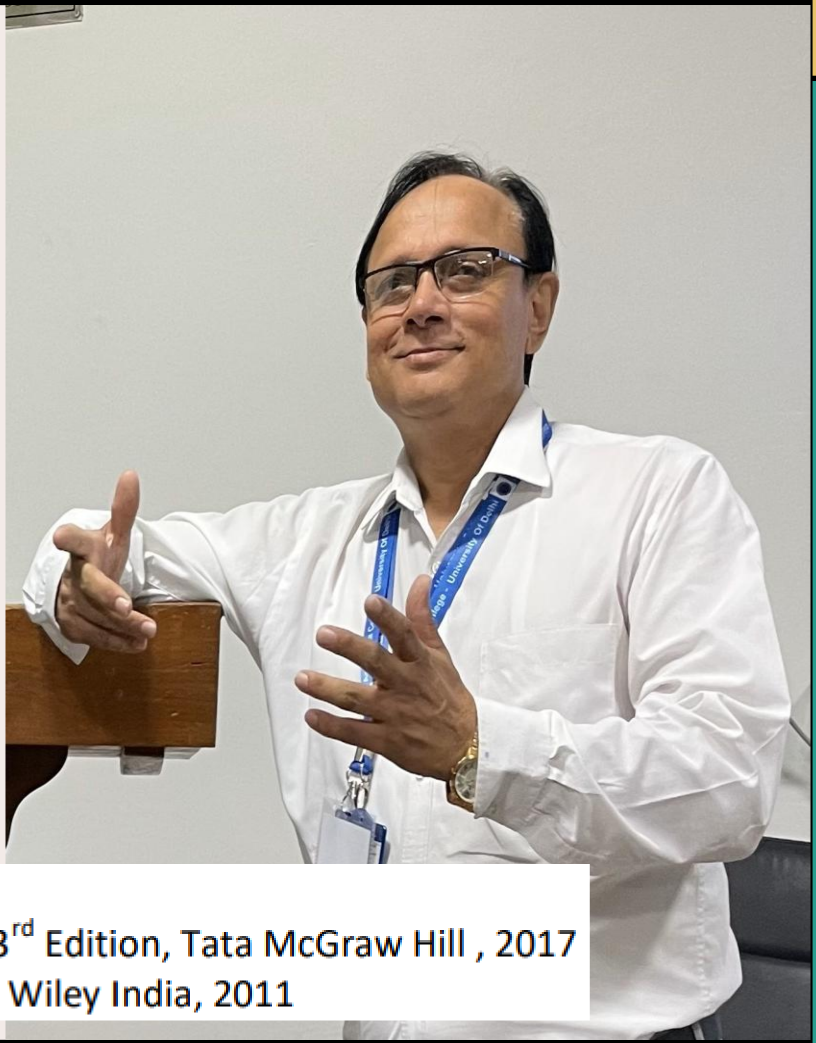
AI and ML
UNIT 2
Paper Code: IT608



Speaker: *Dr Raj Kumar Sharma*

Text:

1. Elaine Rich, Kevin Knight and Shivashankar B Nair, “Artificial Intelligence”, 3rd Edition, Tata McGraw Hill , 2017
2. S. N. Sivanandam, S. N. Deepa, “Principles of Soft Computing”, 2nd Edition, Wiley India, 2011

References: IGNOU study material.

AI and ML Unit 2

Knowledge Representation:
- Representations and Mappings
- Approaches and Issues in Knowledge Representation.
- Using Predicate Logic Rules

Symbolic Reasoning under Uncertainty:
- Nonmonotonic reasoning.

Statistical Reasoning:
- probability and Bayes theorem
- certainty factors and rule-based systems
- Bayesian networks
- Dempster-Shafer theory
- Weak slot-and-filler structures
- Strong slot-and-filler structures.

Knowledge Representation: Representations and Mappings, Approaches and Issues in Knowledge Representation.
Using Predicate Logic, Rules, Symbolic Reasoning under Uncertainty: Nonmonotonic reasoning. Statistical
Reasoning: probability and Bayes theorem, certainty factors and rule-based systems, Bayesian networks,
Dempster-Shafer theory. Weak slot-and-filler structures, Strong slot-and-filler structures.

AI and ML Unit 2

Knowledge Representation: Representations and Mappings Approaches and Issues in Knowledge Representation. Using Predicate Logic Rules

4.1 REPRESENTATIONS AND MAPPINGS

In order to solve the complex problems encountered in artificial intelligence, one needs both a large amount of knowledge and some mechanisms for manipulating that knowledge to create solutions to new problems. A variety of ways of representing knowledge (facts) have been exploited in AI programs. But before we can talk about them individually, we must consider the following point that pertains to all discussions of representation, namely that we are dealing with two different kinds of entities:

- Facts: truths in some relevant world. These are the things we want to represent.
- Representations of facts in some chosen formalism. These are the things we will actually be able to manipulate.

One way to think of structuring these entities is as two levels:

- The *knowledge level*, at which facts (including each agent's behaviors and current goals) are described.
- The *symbol level*, at which representations of objects at the knowledge level are defined in terms of symbols that can be manipulated by programs.

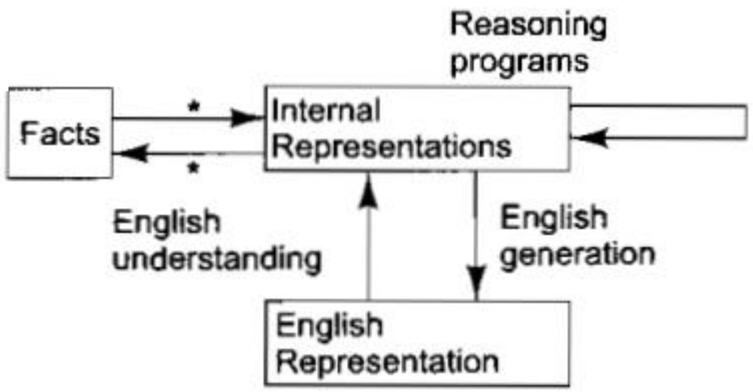


Fig. 4.1 *Mappings between Facts and Representations*

Let's look at a simple example using mathematical logic as the representational formalism. Consider the English sentence:

Spot is a dog.

The fact represented by that English sentence can also be represented in logic as:

$dog(Spot)$

Suppose that we also have a logical representation of the fact that all dogs have tails:

$\forall x : dog(x) \rightarrow hastail(x)$

Then, using the deductive mechanisms of logic, we may generate the new representation object:

$hastail(Spot)$

Using an appropriate backward mapping function, we could then generate the English sentence:

Spot has a tail.

Inheritable Knowledge

The relational knowledge of Fig. 4.4 corresponds to a set of attributes and associated values that together describe the objects of the knowledge base. Knowledge about objects, their attributes, and their values need not be as simple as that shown in our example. In particular, it is possible to augment the basic representation with inference mechanisms that operate on the structure of the representation. For this to be effective, the structure must be designed to correspond to the inference mechanisms that are desired. One of the most useful forms of inference is *property inheritance*, in which elements of specific classes inherit attributes and values from more general classes in which they are included.

In order to support property inheritance, objects must be organized into classes and classes must be arranged in a generalization hierarchy. Figure 4.5 shows some additional baseball knowledge inserted into a structure that is so arranged. Lines represent attributes. Boxed nodes represent objects and values of attributes of objects. These values can also be viewed as objects with attributes and values, and so on. The arrows on the lines point from an object to its value along the corresponding attribute line, The structure shown in the figure is a *slot-and-filler structure*. It may also be called a *semantic network* or a collection *of frames*. In the latter case, each individual frame represents the collection of attributes and values associated with a particular node. Figure 4.6 shows the node for baseball player displayed as a frame.

AI and ML Unit 2

Knowledge Representation: Representations and Mappings Approaches and Issues in Knowledge Representation. Using Predicate Logic Rules

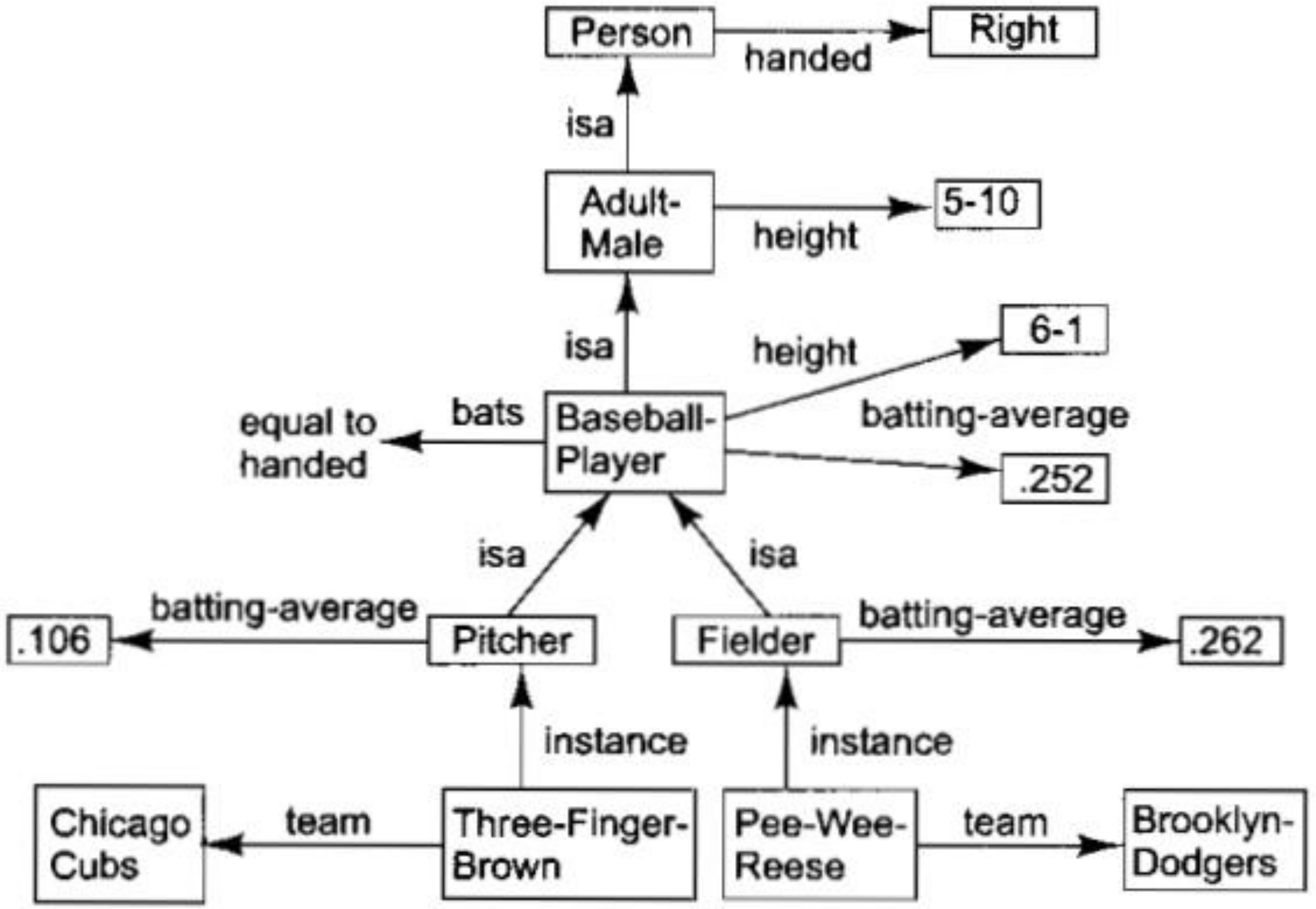


Figure 4.5 *Inheritable Knowledge*

Algorithm: Property Inheritance

To retrieve a value *V* for attribute A of an instance object *O:*

1. Find *O* in the knowledge base.
2. If there is a value there for the attribute A, report that value.
3. Otherwise, see if there is a value for the attribute *instance*. If not, then fail.
4. Otherwise, move to the node corresponding to that value and look for a value for the attribute *A*. If one is found, report it.
5. Otherwise, do until there is no value for the *isa* attrihute or until an answer is found:
   (a) Get the value of the *isa* attribute and move to that node.
   (b) See if there is a value for the attribute *A*. If there is, report it.

Inferential Knowledge

Property inheritance is a powerful form of inference, but it is not the only useful form. Sometimes all the power of traditional logic (and sometimes even more than that) is necessary to describe the inferences that are needed. Figure 4.7 shows two examples of the use of first-order predicate logic to represent additional knowledge about baseball.

$$\forall x : Ball(x) \wedge Fly(x) \wedge Fair(x) \wedge Infield\text{-}Catchable(x) \wedge$$
$$Occupied\text{-}Base(First) \wedge Occupied\text{-}Base(Second) \wedge (Outs < 2) \wedge$$
$$\neg[Line\text{-}Drive(x) \vee Attempted\text{-}Bt,(x)]$$
$$\rightarrow Infield\text{-}Fly(x)$$

$$\forall x,y : Batter(x) \wedge batted(x, y) \wedge Infield\text{-}Fly(y) \rightarrow \mathrm{Out}(x)$$

Fig. 4.7 *Inferential Knowledge*

First-Order Predicate Logic

In Artificial Intelligence, we first encounter the concept of propositions,Now, it's essential to explore the distinction between **propositions** and **predicates** (also referred to as propositional functions).

- A **proposition** is a specific statement that has a definite truth value (true or false).
- A **predicate**, on the other hand, is a generalized statement that involves variables and becomes a proposition only when specific values are assigned to those variables.

Key Differences Between Proposition and Predicate:

1. **Propositions** use only logical connectives (e.g., AND, OR, NOT).
2. **Predicates** use logical connectives as well as **quantifiers**:
    - The **existential quantifier** (denoted as ∃) means "there exists."
    - The **universal quantifier** (denoted as ∀) means "for all."

Understanding Predicates with an Example:

A **propositional function** or **predicate** is a sentence involving a variable (e.g., $p(x)$) that only becomes a proposition when the variable is assigned a specific value. The domain of possible values for the variable is referred to as the **universe of discourse**.

For instance, let's consider $p(x)$:

- $p(x)$ = "x > 5"

In its current form, $p(x)$ is **not a proposition** because it depends on the value of x. However, when x is assigned a specific value:

- If $x = 6$, then $p(6)$ becomes the statement "6 > 5," which is **true**.
- If $x = 0$, then $p(0)$ becomes the statement "0 > 5," which is **false**.

Importance in Predicate Logic:

Predicates allow us to work with more generalized statements. Using quantifiers, we can express statements about entire sets or subsets of the universe of discourse:

- **Existential Quantifier** ($\exists$): "There exists an x such that $p(x)$."
  Example: "There exists a number greater than 5."
- **Universal Quantifier** ($\forall$): "For all x, $p(x)$."
  Example: "All numbers greater than 5 are positive."

4.3 ISSUES IN KNOWLEDGE REPRESENTATION

Before embarking on a discussion of specific mechanisms that have been used to represent various kinds of real-world knowledge, we need briefly to discuss several issues that cut across all of them:

- Are any attributes of objects so basic that they occur in almost every problem domain? If there are, we need to make sure that they are handled appropriately in each of the mechanisms we propose. If such attributes exist, what are they?
- Are there any important relationships that exist among attributes of objects?
- At what level should knowledge be represented? Is there a good set of *primitives* into which all knowledge can be broken down? Is it helpful to use such primitives?
- How should sets of objects be represented?
- Given a large amount of knowledge stored in a database, how can relevant parts be accessed when they are needed?

John spotted Sue.

We could represent this as[1]

spotted(agent(John),
object(Sue))

Such a representation would make it easy to answer questions such as:

Who spotted Sue?

But now suppose we want to know:

Did John see Sue?

The obvious answer is "yes," but given only the one fact we have, we cannot discover that answer. We could, of course, add other facts, such as

$spotted(x, y) \rightarrow saw(x, y)$

We could then infer the answer to the question.

An alternative solution to this problem is to represent the fact that spotting is really a special type of seeing explicitly in the representation of the fact. We might write something such as

saw(agent(John),
object(Sue),
timespan(briefly))

AI and ML Unit 2

Knowledge Representation: Representations and Mappings Approaches and Issues in Knowledge Representation. Using Predicate Logic Rules

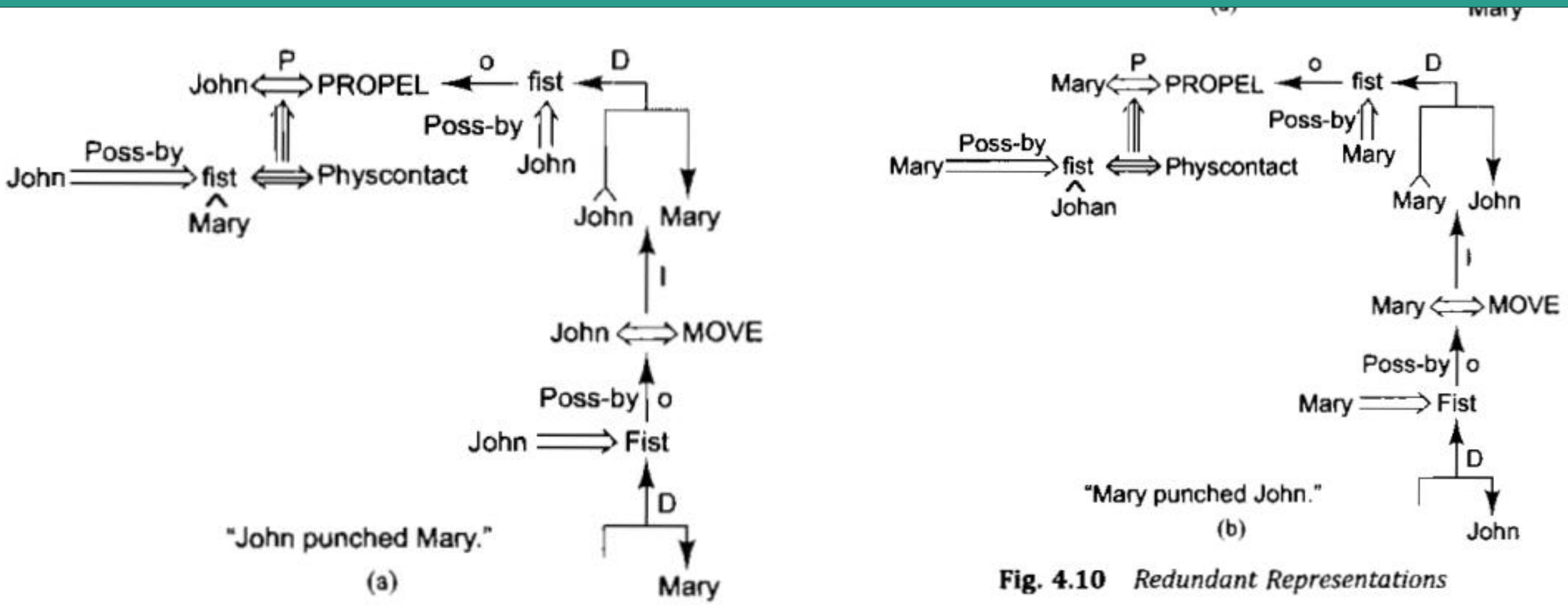


Fig. 4.10 *Redundant Representations*

- Mary = *daughter(brother(mother(Sue)))*
- Mary = *daughter(sister(mother(Sue)))*
- Mary = *daughter(brother(father(Sue)))*
- Mary = *daughter(sister(father(Sue)))*

If we do not already know that Mary is female, then of course there are four more possibilities as well. Since in general we may have no way of choosing among these representations, we have no choice but to represent the fact using the nonprimitive relation *cousin*.

The other way to solve this problem is to change our primitives. We could use the set: *parent, child, sibling, male,* and *female*. Then the fact that Mary is Sue's cousin could be represented as

Mary = *child(sibling(parent(Sue)))*

But now the primitives incorporate some generalizations that may or may not be appropriate. The main point to be learned from this example is that even in very simple domains, the correct set of primitives is not obvious.

In less well-structured domains, even more problems arise. For example, given just the fact

John broke the window.

a program would not be able to decide if John's actions consisted of the primitive sequence:

1. Pick up a hard object.
2. Hurl the object through the window.

or the sequence:

1. Pick up a hard object.
2. Hold onto the object while causing it to crash into the window.

Understanding First-Order Logic: Predicates and Quantifiers

If $q(x)$ is the statement "x has gone to Patna," substituting x with "Taj Mahal" results in a **false proposition**. This illustrates that a predicate, by itself, is typically **not a proposition**; it becomes one only when a specific value is assigned to its variable.

Predicates and Propositions

Every proposition can be considered a special case of a propositional function, similar to how every real number can be viewed as a real-valued constant function. However, not every predicate is a proposition unless its variables are defined with specific values.

Logical Connectives and Quantifiers

Can all sentences be expressed using only logical connectives like AND (∧), OR (∨), and NOT (¬)? Consider a sentence like:

- "x is prime, and $x + 1$ is prime for some x."

Here, the phrase "for some x" introduces a **quantifier** that cannot be expressed with just logical connectives. This is where quantifiers like **existential quantifier** (∃) and **universal quantifier** (∀) come into play:

- ∃: Denotes "there exists."
- ∀: Denotes "for all."

For instance, the sentence "There is at least one child in the class" can be rewritten as:

- $(\exists x \in U)p(x)$, where $p(x)$ means "x is in the class," and U represents the set of all children.

Negating Quantified Statements

Now, consider the negation of the statement above:

- "There is no child in the class."
  This can be symbolized as $(\forall x \in U)q(x)$, where $q(x)$ denotes "x is not in the class" ($q(x) \equiv \neg p(x)$).

In general, the relationship between ∃ and ∀ under negation can be summarized as:

1. $\neg(\forall x \in U)p(x) \equiv (\exists x \in U)(\neg p(x))$.
2. $\neg(\exists x \in U)p(x) \equiv (\forall x \in U)(\neg p(x))$.

Examples of Quantifier Usage

1. **Existential Quantifier** (∃):
   - $(\exists x \in R)(x+1>0)$: "There exists an x in R such that $x+1>0$" (True).
   - $(\exists x \in N)(x-2=0)$: "There exists an x in N such that $x-2=0$" (False).
2. **Universal Quantifier** (∀):
   - $(\forall x \notin N)(x^2>x)$: "For every x not in N, $x^2>x$" (False, as there are counterexamples).

Logical Equivalence Rules

The equivalence between ∀ and ∃ through negation is fundamental in predicate logic:

- $(\forall x \in U)p(x) \equiv \neg(\exists x \in U)(\neg p(x))$.
- $\neg(\forall x \in U)p(x) \equiv (\exists x \in U)(\neg p(x))$.

Here, U represents the **universe of discourse**, the set of values that x can take. These relationships form the basis for reasoning in first-order logic.

To understand how First-Order Predicate Logic (FOPL) extends propositional logic, let's revisit the classic argument:

1. Every man is mortal.
2. Raman is a man.
3. Therefore, Raman is mortal.

Instead of treating each statement as an indivisible atomic unit, FOPL allows us to break them down into **subjects** and **predicates**. In this argument, the two predicates are:

- "is mortal"
- "is a man"

Let's assign the following notations to these predicates:

- $IL(x)$: "x is mortal"
- $IN(x)$: "x is a man"

Using this notation, we can rewrite the argument as follows:

1. $\forall x$, if $IN(x)$, then $IL(x)$ (i.e., all men are mortal).
2. $IN(\text{Raman})$ (i.e., Raman is a man).
3. Therefore, $IL(\text{Raman})$ (i.e., Raman is mortal).

Function symbols can also be incorporated into First-Order Predicate Logic (FOPL). For instance, we can use **product(x, y)** to represent $x \times y$, and **father(x)** to denote "the father of x." Consider the statement: "Mohan's father loves Mohan." This can be symbolized as **LOVE(father(Mohan), Mohan)**. Here, the function **father** allows us to refer to Mohan's father without knowing his name, illustrating the utility of function symbols in such contexts.

In this example:

- Statements like **LIKE(Ram, Mohan)** and **LOVE(father(Mohan), Mohan)** are considered atomic or "atoms" in FOPL. These statements can be assigned a truth value (True or False) and do not involve any logical operators such as ~, ∧, ∨, →, or ↔.

To summarize, from this discussion:

- **LIKE(Ram, Mohan)** and **LOVE(father(Mohan), Mohan)** are atomic statements.
- **GREATER**, **LOVE**, and **LIKE** are **predicate symbols**.
- x and y are **variables**, while 3, Ram, and Mohan are **constants**.
- **father** and **product** are **function symbols**.

Key Concepts of Symbols in FOPL:

1. **Individual symbols or constant symbols**: Represent specific objects, such as names (Ram, Mohan) or numbers (3, 5).
2. **Variable symbols**: Typically lowercase letters like x, y, z, or subscripted forms like x_3.
3. **Function symbols**: Usually lowercase letters like f, g, h, or descriptive strings such as **father** or **product**.
4. **Predicate symbols**: Represent relationships or properties, denoted by uppercase letters like P, Q, R, or strings such as **greater_than** or **is_tall**.

Function and Predicate Arity:

- A function symbol that takes n arguments is called an **n-place function symbol**.
- Similarly, a predicate symbol that takes m arguments is called an **m-place predicate symbol**.
  - For example, **father** is a one-place function symbol.
  - **GREATER** and **LIKE** are two-place predicate symbols.
  - **father_of(x, y)** is a two-place predicate symbol.

Terms in FOPL:

A **term** refers to the symbolic representation of a function or predicate argument. It is defined recursively as follows:

1. A **variable** is a term.
2. A **constant** is a term.
3. If f is an n-place function symbol and $t_1, t_2, \ldots, t_n$ are terms, then $f(t_1, t_2, \ldots, t_n)$ is also a term.
4. Any term can be constructed using these rules.

This systematic approach to terms, constants, variables, predicates, and functions forms the foundation of First-Order Predicate Logic.

Notes on First-Order Predicate Logic (FOPL)

1. **Terms and Function Symbols**:
   - A term is a symbol representing an object.
   - Example: "y" and "3" are terms.
   - A two-place function symbol (like "plus") applied to terms results in a term:
     - Example: "plus(y, 3)" is a term.
   - Nested terms are also valid:
     - Example: "plus(plus(y, 3), y)" is a term, representing "(y + 3) + y".
     - "father(father(Mohan))" is also a term, representing the grandfather of Mohan.

2. **Predicates**:
   - A predicate is a function that maps a list of constant arguments to either True (T) or False (F).
   - Example: "GREATER(5, 2)" is True (T), "GREATER(1, 3)" is False (F).
3. **Atoms**:
   - In Propositional Logic (PL), an atom is an indivisible unit (e.g., symbols like P, Q, R).
   - In First-Order Predicate Logic (FOPL):
     - An atom can be:
       1. An atom from Propositional Logic, or
       2. A predicate applied to terms.
     - Example: "P(t1, t2, ..., tn)" is an atom if "P" is an n-place predicate and "t1, t2, ..., tn" are terms.

4. **Building Complex Formulas**:
   - Using logical connectives from Propositional Logic (with the same meaning in FOPL), complex formulas can be built.
   - Logical connectives: AND, OR, NOT, etc.
5. **Quantifiers**:
   - **Universal Quantifier (∀)**: Represents "for all".
     - Example: (∀x) Q(x) means "for all x, Q(x) is true".
   - **Existential Quantifier (∃)**: Represents "there exists".
     - Example: (∃x) Q(x) means "there exists an x such that Q(x) is true".

6. **Example Symbolizations**:
   - Statement 1: "There exists a number that is rational."
     - Symbolization: $(\exists x)\ Q(x)$, where $Q(x)$ represents "x is a rational number."
   - Statement 2: "Every rational number is a real number."
     - Symbolization: $(\forall x)\ (Q(x) \rightarrow R(x))$, where $R(x)$ represents "x is a real number."
   - Statement 3: "For every number x, there exists a number y such that y is greater than x."
     - Symbolization: $(\forall x)\ (\exists y)$ LESS(x, y), where LESS(x, y) means "x is less than y."
7. **Well-Formed Formulas (wff)**:
   - A formula or well-formed formula (wff) is a valid expression in FOPL.
   - Examples of well-formed formulas:
     - (i) $(\exists x)\ Q(x)$
     - (ii) $(\forall x)\ (Q(x) \rightarrow R(x))$
     - (iii) $(\forall x)\ (\exists y)$ LESS(x, y)

$\sim(\forall\ x \in U)p(x) \equiv \sim\sim(\exists\ x \in U)(\sim p(x)) \equiv (\exists\ x \in U)(\sim p(x))$.

This is one of the rules for negation that relate $\forall$ and $\in$. The two rules are

$\sim(\forall\ x \in U)p(x) \equiv (\exists\ x \in U)(\sim p(x))$, and

$\sim(\exists\ x \in U)p(x) \equiv (\forall\ x \in U)(\sim p(x))$

Where U is the set of values that x can take.

- **Open and Closed Formulas**:
  - Formulas like "greater(x, y)", "greater(x, 3)", and "∀y greater(x, y)" are open formulas because they contain a free occurrence of the variable **x**.
  - On the other hand, formulas like "(∀x) (∃y) greater(x, y)", "(∀y) greater(y, 1)", and "greater(9, 2)" are closed formulas since they do not contain any free variables.
- **Logical Equivalences**:
  - (i) **(∀x) P(x) ∧ (∀x) Q(x) = (∀x) (P(x) ∧ Q(x))**: If both **P(x)** and **Q(x)** hold for all **x**, then both **P(x)** and **Q(x)** must hold together for every **x**.
  - (ii) **(∃x) P(x) ∨ (∃x) Q(x) = (∃x) (P(x) ∨ Q(x))**: If there exists an **x** where **P(x)** is true or an **x** where **Q(x)** is true, then there exists an **x** such that either **P(x)** or **Q(x)** is true.

- **Logical Inequalities**:
  - (iii) **$(\forall x)\ (P(x) \lor Q(x)) \neq (\forall x)\ P(x) \lor (\forall x)\ Q(x)$**: The left-hand side correctly states that every **x** is either **P(x)** or **Q(x)**, but the right-hand side incorrectly states that every **x** is either **P(x)** or **Q(x)** exclusively.
  - (iv) **$(\exists x)\ (P(x) \land Q(x)) \neq (\exists x)\ P(x) \land (\exists x)\ Q(x)$**: The left-hand side is incorrect, as no **x** can be both **P(x)** and **Q(x)** (odd and even), while the right-hand side is correct as it states there is an odd number and an even number.
- **Negation of Quantifiers**:
  - (v) $\sim(\forall x)\ P(x) = (\exists x)\ \sim P(x)$: Negating "for all **x**, **P(x)**" is the same as stating that there exists an **x** such that **P(x)** is false.
  - (vi) $\sim(\exists x)\ P(x) = (\forall x)\ \sim P(x)$: Negating "there exists an **x** such that **P(x)**" is equivalent to saying "for all **x**, **P(x)** is false".

(i) Universal Instantiation Rule (U.I.):

$$\frac{(\forall x)\,p(x)}{p(a)}$$

Where is an a arbitrary constant.

The rule states if ***(∀x) p(x)*** is True, then we can assume ***P(a)*** as True for any constant ***a*** (where a constant ***a*** is like Raman). It can be easily seen that the rule associates a formula ***P(a) of Propositional Logic*** to a formula ***(∀x) p(x) of FOPL.*** The significance of the rule lies in the fact that once we obtain a formula like ***P(a)***, then the reasoning process of Propositional Logic may be used. The rule may be used , whenever, its application seems to be appropriate.

(ii) Universal Generalisation Rule (U.G.)

$$\frac{P(a),\ for\ all\ a}{(\forall x)\, p(x)}$$

The rule says that if it is known that for all constants a, the statement P(a) is True, then we can, instead, use the formula $(\forall x)\, p(x)$.

The rule associates with a set of formulas **P(a) for all a of Propositional Logic,** a formula $(\forall x)\, p(x)$ of FOPL.

Before using the rule, we must ensure that P(a) is True for all a, Otherwise it may lead to wrong conclusions.

(iii) Existential Instantiation Rule (E. I.)

$$\frac{(\exists x)\,P(x)}{P(a)} \qquad (E.I.)$$

The rule says if the Truth of $(\exists x)\ P(x)$ *is known then we can assume the Truth of P(a) for* ***some fixed a****. The rule, again, associates a formula P(a) of Propositional Logic to a formula* $(\forall x)\,p(x)$ *of FOPL.*

An inappropriate application of this rule may lead to wrong conclusions. The source of possible errors lies in the fact that the choice 'a' in the rule is not arbitrary and can not be known at the time of deducing P(a) from $(\exists x)\ P(x)$.

If during the process of deduction some other $(\exists y)\ Q(y)$ or $(\exists x)\ (R(x))$ or even another (∃x)P(x) is encountered, then each time a new constant say *b*, *c* etc. should be chosen to infer Q (b) from $(\exists y)\ Q(y)$ or R(c) from $(\exists x)\,(R(x)$) or P(d) from $(\exists x)\ P(x)$.

(iv) Existential Generalization Rule (E.G)

$$\frac{P(a)}{(\exists x)P(x)} \qquad \text{(E.G)}$$

The rule states that if P(a), a formula of Propositional Logic is True, then the Truth of , a formula of FOPL , may be assumed to be True.

The Universal Generalisation (U.G) and Existential Instantiation rules should be applied with utmost care, however, other two rules may be applied, whenever, it appears to be appropriate.

Next, The purpose of the two rules, viz.,

(i) Universal Instantiation Rule (U. I.)

(iii) Existantial Rule (E. I.)

Given Inference:

- **To conclude**: $F(a) \wedge G(a) \rightarrow H(a) \wedge I(a)$
- **From**: $(\forall x)(F(x) \wedge G(x)) \rightarrow H(x) \wedge I(x)$
- **Using**: Universal Instantiation (U.I.)

Analysis:

1. **Universal Quantification**: The given statement $(\forall x)(F(x) \wedge G(x)) \rightarrow H(x) \wedge I(x)$ is a universally quantified formula, meaning it applies to **every** x. In simpler terms, "For all x, if $F(x) \wedge G(x)$ holds, then $H(x) \wedge I(x)$ must also hold."
2. **Universal Instantiation (U.I.)**: The rule allows us to instantiate the universally quantified variable (x) with a specific constant. In this case, we are trying to instantiate it with the constant a. The general formula $(\forall x)(F(x) \wedge G(x)) \rightarrow H(x) \wedge I(x)$ should then yield the formula $(F(a) \wedge G(a)) \rightarrow H(a) \wedge I(a)$ upon instantiating x with a.

AI and ML Unit 2

Knowledge Representation: Representations and Mappings Approaches and Issues in Knowledge Representation. Using Predicate Logic Rules

Example: Symbolize the following and then construct a proof for the argument:

(i) Anyone who repairs his own car is highly skilled and saves a lot of money on repairs

(ii) Some people who repair their own cars have menial jobs. Therefore,

(iii) Some people with menial jobs are highly skilled.

Solution: Let us use the notation:

P(x) : x is a person

S(x) : x saves money on repairs

M(x) : x has a menial job

R(x) : x repairs his own car

H(x) : x is highly skilled.

Therefore, (i), (ii) and (iii) can be symbolized as:

(i) $(\forall x)\ (R(x)\ \ (H(x) \wedge S(x)))$

(ii) $\exists(x)\ (R(x) \wedge M(x))$

(iii) $(\exists x)\ (M(x) \wedge H(x))$ (to be concluded)

From (ii) using Existential Instantiation (E.I), we get, for some fixed a

(iv) $R(a) \wedge M(a)$

Then by simplification rule of Propositional Logic, we get

(v) $R(a)$

From (i), using Universal Instantiation (U.I.), we get

(vi) $R(a) \rightarrow H(a) \wedge S(a)$

Using modus ponens w.r.t. (v) and (vi) we get

(vii) $H(a) \wedge S(a)$

By specialisation of (vii) we get

(viii) $H(a)$

By specialisation of (iv) we get

(ix) $M(a)$

By conjunctions of (viii) & (ix) we get

$M(a) \wedge H(a)$

By Existential Generalisation, we get

$(\exists x)\ (M(x) \wedge H(x))$

Hence, (iii) is concluded.

Prenex Normal Form (PNF) is a way of expressing logical formulas (particularly in predicate logic) where all quantifiers (such as ∀ for "for all" and ∃ for "there exists") are moved to the front of the formula, followed by a quantifier-free matrix (the part of the formula that remains after the quantifiers are moved). The purpose of Prenex Normal Form is to standardize logical expressions, making them easier to manipulate for automated theorem proving and logic analysis.

Key Concepts:

1. **Quantifiers**: Quantifiers like ∀ (for all) and ∃ (there exists) apply to variables in predicate logic.
2. **Matrix**: The matrix is the part of the formula that does not contain any quantifiers. It consists of the logical connectives (AND, OR, NOT, etc.) and predicates.
3. **Prenex Form**: The formula is in Prenex Normal Form if it consists of a string of quantifiers followed by the matrix. The quantifiers precede the matrix.

Structure of Prenex Normal Form:

A formula in Prenex Normal Form has the following structure:

$$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \, \phi(x_1, x_2, \ldots, x_n)$$

Where:

- $Q_1, Q_2, \ldots, Q_n$ are quantifiers (either ∀ or ∃).
- $x_1, x_2, \ldots, x_n$ are the variables.
- $\phi(x_1, x_2, \ldots, x_n)$ is the matrix, a quantifier-free part of the formula.

Steps to Convert a Formula into Prenex Normal Form:

1. **Move Quantifiers to the Front**:
   - If quantifiers are not at the front of the formula, they need to be moved. However, care must be taken to preserve the meaning of the formula. This may involve using **quantifier rules** such as **quantifier exchange**, **variable renaming**, and **quantifier negation**.
2. **Skolemization** (for existential quantifiers):
   - If there are existential quantifiers in the formula, we may need to replace them with Skolem functions or constants to eliminate the existential quantifier, especially when dealing with universal quantifiers. This step is crucial when converting to **Skolem normal form**, which is related to Prenex Normal Form.
3. **Remove Quantifiers from the Matrix**:
   - The goal is to rewrite the formula so that all the quantifiers come first, followed by a quantifier-free matrix. To do this, **logical equivalences** (like distributive properties) are used to eliminate quantifiers from the matrix.
4. **Apply Logical Equivalences**:
   - Logical equivalences like De Morgan's Laws and distributive properties help transform the formula while keeping the logical meaning intact.

Example 1: Converting a Formula into Prenex Normal Form

Given the formula:

$$\forall x\,(P(x) \rightarrow \exists y\,Q(x,y))$$

Step 1: Eliminate the implication:

$$\forall x\,(\neg P(x) \vee \exists y\,Q(x,y))$$

Step 2: Move the existential quantifier outside:

$$\forall x\,\exists y\,(\neg P(x) \vee Q(x,y))$$

Now, the formula is in **Prenex Normal Form**, where all quantifiers are at the front, and the matrix $\neg P(x) \vee Q(x,y)$ follows.

Example 2: Formula with Multiple Quantifiers

Consider the formula:

$$\exists x\,\forall y\,(P(x,y) \vee Q(y))$$

Step 1: Move quantifiers to the front (if needed). In this case, the quantifiers are already in the correct order:

$$\exists x\,\forall y\,(P(x,y) \vee Q(y))$$

Step 2: The formula is already in **Prenex Normal Form**, since the quantifiers come first and the matrix $P(x,y) \vee Q(y)$ follows.

Why is Prenex Normal Form Useful?

- **Simplification for Automated Theorem Proving**: Converting a formula into Prenex Normal Form makes it easier for automated systems to apply inference rules, particularly when dealing with complex logical expressions involving multiple quantifiers.
- **Standardized Representation**: It provides a standardized way to represent formulas, making it easier to compare and manipulate formulas logically.
- **Compatibility with Other Forms**: Many logical algorithms and proofs work more effectively with formulas in Prenex Normal Form because it allows easier manipulation and application of logical rules.

Examples of Formulas in Prenex Normal Form:

1. $(\exists x)(\forall y)(R(x,y) \vee Q(y))$
   - Here, the quantifiers are at the beginning, followed by the matrix $R(x,y) \vee Q(y)$, which is free of quantifiers.
2. $(\forall x)(\forall y)(\neg P(x,y) \to S(y))$
   - This formula has the prefix $(\forall x)(\forall y)$ and the matrix $\neg P(x,y) \to S(y)$.
3. $(\forall x)(\forall y)(\exists z)(P(x,y) \to R(z))$
   - Similarly, this formula has quantifiers at the beginning and a quantifier-free matrix $P(x,y) \to R(z)$.

Then, the following laws involving quantifiers hold good in FOPL

(i) $(Qx)\, P[x] \vee G = (Qx)\,(P[x] \vee G)$.

(ii) $(Qx)\, P[x] \wedge G = (Qx)\,(P[x] \wedge G)$.

In the above two formulas, Q may be either $\forall$ or $\exists$.

(iii) $\sim((\forall x)\, P[x]) = (\exists x)\,(\sim P[x])$.

(iv) $\sim((\exists x)\, P[x]) = (\forall x)\,(\sim P[x])$.

(v) $(\forall x)\, P[x] \wedge (\forall x)\, H[x] = (\forall x)\,(P[x] \wedge H[x])$.

(vi) $(\exists x)\, P[x] \vee (\exists x)\, H[x] = (\exists x)\,(P[x] \vee H[x])$.

That is, the universal quantifier $\forall$ and the existential quantifier $\exists$ can be distributed respectively over $\wedge$ and $\vee$.

But we must be careful about (we have already mentioned these inequalities)

(vii) $(\forall x)\, E[x] \vee (\forall x)\, H[x] \neq (\forall x)\,(P[x] \vee H[x])$ and

(viii) $(\exists x)\, P[x] \wedge (\exists x)\, H[x] \neq (\exists x)\,(P[x] \wedge H[x])$

Steps for Transforming an FOPL Formula into Prenex Normal Form

Step 1 Remove the connectives '$\leftrightarrow$' and '$\rightarrow$' using the equivalences

$P \leftrightarrow G = (P \rightarrow G) \wedge (G \rightarrow P)$

$P \rightarrow G = \sim P \rightarrow G$

Step 2 Use the equivalence to remove even number of ~'s

$\sim(\sim P) = P$

Step 3 Apply De Morgan's laws in order to bring the negation signs immediately before atoms.

$\sim(P \vee G) = \sim P \wedge \sim G$

$\sim(P \wedge G) = \sim P \vee \sim G$

and the quantification laws

$\sim((\forall x)\, P[x]) = (\exists x)\,(\sim P[x])$

$\sim((\exists x)\, P[x]) = (\forall x)\,(\sim F[x])$

Step 4 rename bound variables if necessary

Step 5 Bring quantifiers to the left before any predicate symbol appears in the formula. This is achieved by using (i) to (vi) discussed above.

AI and ML Unit 2

Knowledge Representation: Representations and Mappings Approaches and Issues in Knowledge Representation. Using Predicate Logic Rules

Example: Transform the following formulas into prenex normal forms:

(i) $(\forall x)\,(Q(x) \to (\exists x)\,R(x, y))$

(ii) $(\exists x)\,(\sim (\exists y)\,Q(x, y) \to ((\exists z)\,R(z) \to S(x)))$

(iii) $(\forall x)\,(\forall y)\,((\exists z)\,Q(z, y, z) \land ((\exists u)\,R(x, u) \to (\exists v)\,R(y, v)))$.

Part (i)

Step 1: *By removing '→', we get*

$(\forall x)\,(\sim Q(x) \lor (\exists x)\,R(x, y))$

Step 2: *By renaming x as z in (∃x) R (x, y) the formula becomes*

$(\forall x)\,(\sim Q(x) \lor (\exists z)\,R(z, y))$

Step 3: *As ~ Q(x) does not involve z, we get*

$(\forall x)\,(\exists z)\,(\sim Q(x) \lor R(z, y))$

Part (ii)

$(\exists x)\,(\sim (\exists y)\,Q(x, y) \to ((\exists z)\,R(z) \to S(x)))$

Step 1: *Removing outer '□' we get*

$(\exists x)\,(\sim (\sim ((\exists y)\,Q(x, y))) \lor ((\,\square z)\,R(z)\,\square\,S(x)))$

Step 2: *Removing inner '→', and simplifying ~ (~ ( )) we get*

$(\exists x)\,((\exists y)\,Q(x, y) \lor (\sim ((\exists z)\,R(z)) \to S(x)))$

Step 3: *Taking '~' inner most, we get*

$(\exists x)\,(\exists y)\,Q(x, y) \lor ((\forall z) \sim R(z) \lor S(x)))$

As first component formula Q (x, y) does not involve z and S(x) does not involve both y and z and ~ R(z) does not involve y. Therefore, we may take out (∃ y) and (∀z) so that, we get

$(\exists x)\,(\exists y)\,(\forall z)\,(Q(x, y) \lor (\sim R(z) \lor S(x))$, which is the required formula in prenex normal form.

Part (iii)

$(\forall x)\ (\forall y)\ ((\exists z)\ Q\ (x, y, z) \wedge ((\exists u)\ R\ (x, u) \rightarrow (\exists v)\ R\ (y\ v)))$

Step 1: Removing '→', we get

$(\forall x)\ (\forall y)\ ((\exists z)\ Q\ (x, y, z) \wedge (\sim ((\forall u)\ R\ (x, u)) \vee (\exists v)\ R\ (y, v)))$

Step 2: Taking '~' inner most, we get

$(\forall x)\ (\forall y)\ ((\exists z)\ Q\ (x, y, z) \wedge ((\forall u) \sim R\ (x, u) \vee (\exists v)\ R\ (y, v)))$

Step 3: As variables z, u & v do not occur in the rest of the formula except the formula which is in its scope, therefore, we can take all quantifiers outside, preserving the order of their occurrences, Thus we get

$(\forall x)\ (\forall y)\ (\exists z)\ (\forall u)\ (\exists v)\ (Q\ (x, y, z) \wedge (\sim R\ (x, u) \vee R\ (y, v)))$

Skolomization: A further refinement of Prenex Normal Form (PNF) called (Skolem) Standard Form, is the basis of problem solving through Resolution Method. The Resolution Method will be discussed next.

The **Standard Form of a formula of FOPL is** obtained through the following three steps:

(1) The given formula should be converted to Prenex Normal Form (PNF), and then

(2) Convert the Matrix of the PNF, i.e, quantifier-free part of the PNF into conjunctive normal form

(3) Skolomization: Eliminate the existential quantifiers using skolem constants and functions

Before illustrating the process of conversion of a formula of FOPL to Standard Normal Form, through examples, we discuss briefly skolem functions.

Skolem Function

We in general, mentioned earlier that $(\exists x)(\forall y) P(x,y) \neq (\forall y)(\exists x) P(x,y)$……. (1)

For example, if P(x,y) stands for the relation 'x>y' in the set of integers, then the L.H.S. of the inequality (i) above states: some (fixed) integer (x) is greater than all integers (y). This statement is False.

On the other hand, R.H.S. of the inequality (1) states: for each integer y, there is an integer x so that x>y. This statement is True.

The difference in meaning of the two sides of the inequality arises because of the fact that on L.H.S. x in $(\exists x)$ is independent of y in $(\forall y)$ **whereas** on R.H.S x of dependent on y. In other words, x on L.H.S. of the inequality can be replaced by some constant say 'c' whereas on the right hand side x is some function, say, f(y) of y.

Therefore, the two parts of the inequality (i) above may be written as

LH.S. of (1) $= (\exists x)(\forall y) P(x,y) = (\forall y) P(c,y)$,

Dropping x because there is no x appearing in $(\forall y) P(c,y)$

R.H.S. of (1) $= (\forall y)(\exists x) P(f(y),y) = (\forall y) P(f(y), y)$

The above argument, in essence, explains what is meant by each of the terms viz. *skolem constant, skolem function and skolomisation.*

The constants and functions which replace existential quantifiers are respectively called **skolem constants and skolem functions**. The process of replacing all existential variables by skolem constants and variables is called **skolemisation**.

A form of a formula which is obtained after applying the steps for

(i) reduction to PNF and then to

(ii) CNF and then

(iii) applying skolomization is called **Skolem Standard Form** or just **Standard Form.**

We explain through examples, the skolomisation process after PNF and CNF have already been obtained.

Example: Skolomize the following:

(i) $(\exists x_1)(\exists x_2)(\forall y_1)(\forall y_2)(\exists x_3)(\forall y_3)\, P(x_1, x_2, x_3, y_1, y_2, y_3)$

(ii) $(\exists x_1)(\forall y_1)(\exists x_2)(\forall y_2)(\exists x_3)P(x_1, x_2, x_3, y_1, y_2) \wedge (\exists x_1)(\forall y_3)(\exists x_2)(\forall y_4)Q(x_1, x_2, y_3, y_4)$

Solution (i) As existential quantifiers x1 and x2 precede all universal quantifiers, therefore, x1 and x2 are to be replaced by constants, but by distinct constants, say by ‘c’ and ‘d’ respectively. As existential variable x3 is preceded by universal quantifiers y1 and y2, therefore, x3 is replaced by some function f(y1, y2) of the variables y1 and y2. After making these substitutions and dropping universal and existential variables, we get the skolemized form of the given formula as

$(\forall y_1)(\forall y_2)(\forall y_3)\,(c, d, f(y_1, y_2), y_1, y_2, y_3)$.

Solution (ii) As a first step we must bring all the quantifications in the beginning of the formula through Prenex Normal Form reduction. Also,

$$(\exists x)\ldots P(x,\ldots) \wedge (\exists x)\ldots Q(x,\ldots) \neq (\exists x)(\ldots P(x) \wedge \ldots Q(x,\ldots),$$

therefore, we rename the second occurrences of quantifiers $(\forall x_1)$ and $(\forall x_2)$ by renaming these as x_5 and x_6. Hence, after renaming and pulling out all the quantifications to the left, we get

$$(\exists x_1)(\forall y_1)(\exists x_2)(\forall y_2)(\exists x_3)(\exists x_5)(\forall y_3)(\exists x_6)(\forall y_4)$$

$$(P(x_1, x_2, x_3, y_1, y_2) \wedge Q(x_5, x_6, y_3, y_4)$$

Then the existential variable x_1 is independent of all the universal quantifiers. Hence, x_1 may be replaced by a constant say, ‘c’. Next x_2 is preceded by the universal quantifier y_1 hence, x_2 may be replaced by $f(y_1)$. The existential quantifier x_3 is preceded by the universal quantifiers y1 and y_2. Hence x_3 may be replaced by g

(y_1, y_2). The existential quantifier x_5 is preceded by again universal quantifier y_1 and y_2. In other words, x_5 is also a function of y_1 and y_2. But, we have to use a different function symbol say h and replace x_5 by $h(y_1, y_2)$. Similarly x_6 may be replaced by

$j(y_1, y_2, y_3)$.

Thus, (Skolem) Standard Form becomes

$(\forall y_1)\ (\forall y_2)\ (\forall y_3)\ (P\ (c, f(y_1), g(y_1, y_2), y_1, y_2) \wedge Q\ (h\ (y_1, y_2), j\ (y_1, y_2, y_3))).$

Check Your Progress -2

Ex: 4 (i) Transform the formula $(\forall x)\ P(x) \rightarrow (\exists x)\ Q(x)$ into prenex normal form.

(ii) Obtain a prenex normal form for the formula

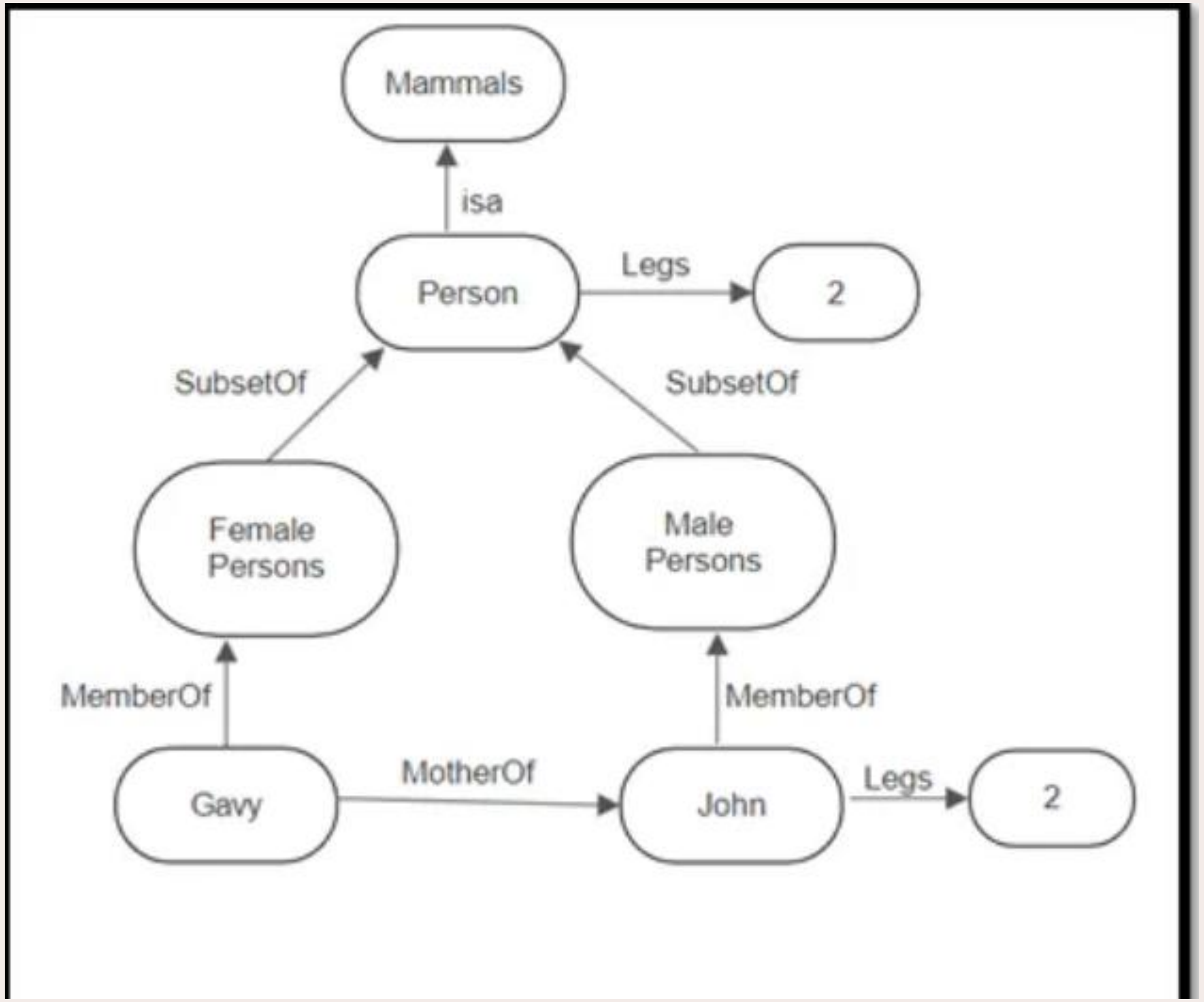
$(\forall x)\ (\forall y)\ ((\exists z)\ (P(x, y) \wedge P(y, z)) \rightarrow (\exists u)\ Q\ (x, y, u))$

Ex 5. Obtain a (skolem) standard form for each of the following formula:

(i) $(\exists x)\ (\forall y)\ (\forall v)\ (\exists z)\ (\forall w)\ (\exists u)\ P\ (x, y, z, u, v, w)$

(ii) $(\forall x)\ (\exists y)\ (\exists z)\ ((P\ (x, y) \vee \sim Q\ (x, z)) \rightarrow R\ (x, y, z))$

Symantec Network



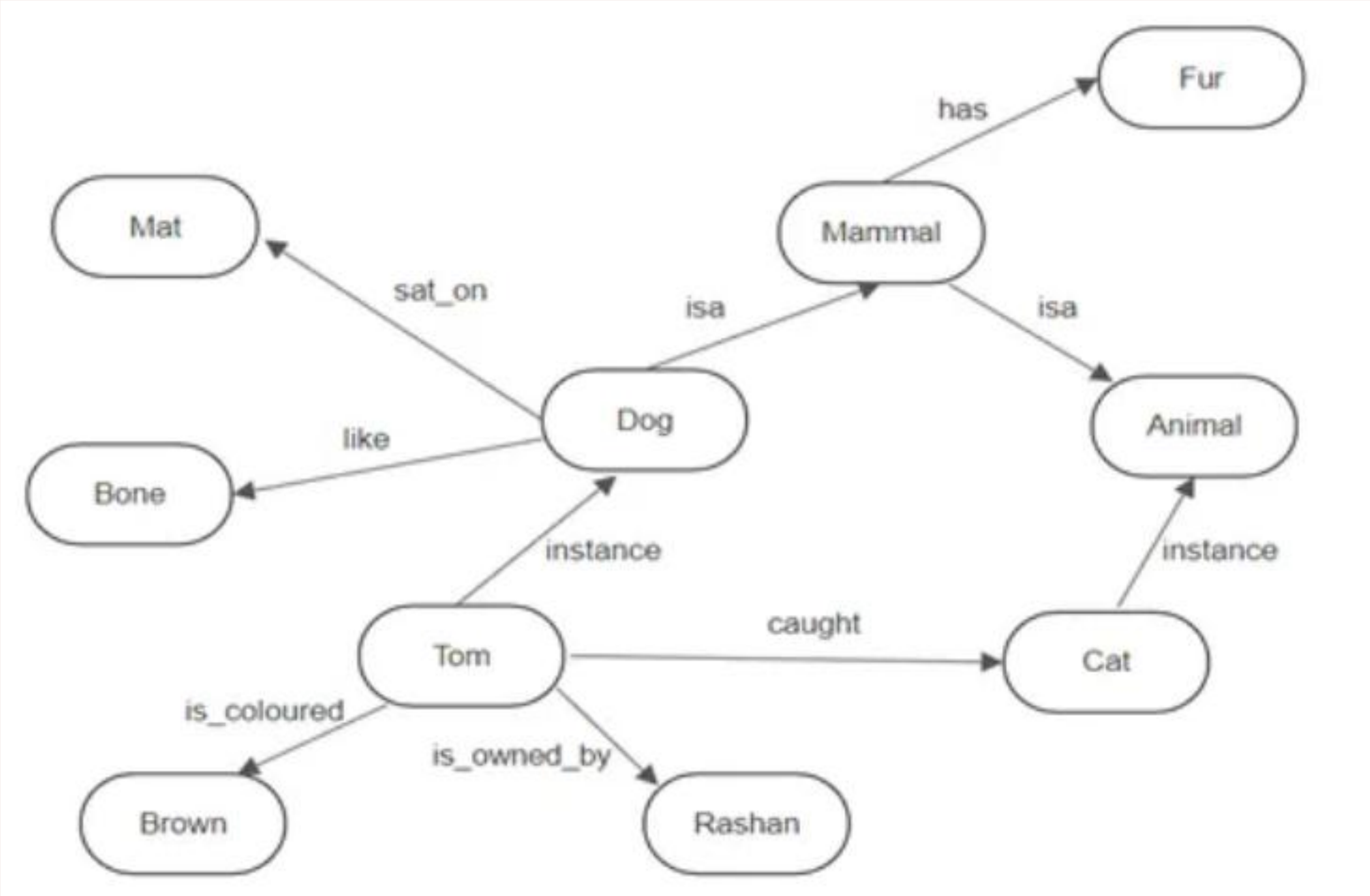
Advantages

- Semantic networks can represent **default values** for categories. For instance, if "John" has one leg but belongs to the "Person" category, which typically has two legs, the default value can be **overridden** by a specific attribute.
- They present information in a **clear and intuitive** way.
- Semantic networks are **simple** and **easy to understand**.
- They can be easily **translated into PROLOG**.

Disadvantages

- A key limitation of semantic networks is that the **relationships between objects are restricted to binary relations**. For example, a statement like *Run(RajdhaniExpress, Chandigarh, Delhi, Tomorrow)* cannot be directly represented.

Symantec Network



Tom is an instance of a **dog**.

- Tom **caught** a cat.
- Tom is **owned by** Rashan.
- Tom is **brown in color**.
- Dogs **like bones**.
- The dog **sat on the mat**.
- A dog is a **mammal**.
- A cat is an **instance of an animal**.
- All **mammals are animals**.
- Mammals **have fur**.

This network demonstrates **"isa"** relationships along with other **domain-specific** relationships such as **sat_on, like, caught, is_coloured, and is_owned_by**. Using **inheritance**, we can infer the additional relation:

sat_on(Tom, Mat).

Concept of Frames

Introduced by **Marvin Minsky in 1974**, the **frame concept** is a fundamental approach in **knowledge representation**. Frames provide a structured way to capture the essential characteristics of a situation, making information **easier to retrieve and manipulate**. They function similarly to **schemas or blueprints**, organizing knowledge into structured units.

Key Components of Frames

Frames play a crucial role in **structuring knowledge in AI**, and understanding their components enhances their effective use.

Key Components of Frames

Frames play a crucial role in **structuring knowledge in AI**, and understanding their components enhances their effective use.

1. Slots

Slots define the **attributes** or **properties** of a frame, representing different aspects of the concept.

Example: A "Person" frame may include the following slots:

- **Name**: The individual's name
- **Age**: The individual's age
- **Occupation**: The individual's profession
- **Address**: The individual's home address

2. Facets

Facets provide **additional constraints or details** for slots, specifying acceptable values or usage rules.

Example: For the **"Age"** slot in a "Person" frame:

- **Type**: Integer
- **Range**: 0 to 120
- **Default Value**: 30

3. Default Values

Default values serve as **predefined values** for slots if no specific data is provided. They establish a baseline that can be modified when needed.

Example: In a "Car" frame:

- **Make**: Default value → "Unknown"
- **Model**: Default value → "Unknown"
- **Year**: Default value → Current year

4. Procedures

Procedures are **functions or methods** linked to a frame that determine how its information is processed or applied.

Example: In an "Account" frame:

- **Procedure:** `CalculateInterest` – Computes interest based on the account balance.

Example: Complete Frame for a Book

Here's how a **"Book"** frame might be structured in a **library management system**:

Frame Name: **Book**

Slots:

- **Title**: "To Kill a Mockingbird"
- **Author**: "Harper Lee"
- **Publication Year**: 1960
- **ISBN**: "978-0-06-112008-4"
- **Genre**: "Fiction"

Facets:

- **Publication Year**:
  - **Type**: Integer
  - **Range**: 1450 to the current year (valid range for book publication)
- **ISBN**:
  - **Format**: 13-digit number

Default Values:

- **Genre**: "Unknown" (if not specified)

Procedures:

- **CheckAvailability**: Determines whether the book is currently available in the library.
- **UpdateRecord**: Updates the book's record when it is borrowed or returned.

Frames provide a powerful way to **structure knowledge**, supporting **efficient reasoning, inheritance, and data retrieval** in AI systems.

Example: "Car" Frame

Here is a simple **frame** representing a **Car**:

Frame Name: **Car**

Slots:

- **Make**: Toyota
- **Model**: Corolla
- **Year**: 2022
- **Color**: White

Facets:

- **Year**:
  - **Type**: Integer
  - **Range**: 1886 to the current year (since the first car was made in 1886)

Default Values:

- **Color**: "Unknown" (if not specified)

Procedures:

- **StartEngine()** – A function to start the car's engine.
- **CheckFuelLevel()** – A function to check the fuel level.

This frame organizes **car-related knowledge** in a structured manner, making it easy to retrieve and update information

Example: Logic-Based Representation

Scenario: Representing the relationship between a **parent** and a **child** using **First-Order Logic (FOL)**.

Facts (Knowledge Representation)

1. **Parent-Child Relationship:**
   - $\text{Parent}(John, Alice)$ → John is a parent of Alice.
   - $\text{Parent}(Mary, Alice)$ → Mary is a parent of Alice.
   - $\text{Parent}(John, Bob)$ → John is a parent of Bob.
2. **Rule (General Knowledge)**
   - If **X** is a parent of **Y**, then **Y** is a child of **X**:

$$\forall X, Y \ (\text{Parent}(X, Y) \rightarrow \text{Child}(Y, X))$$

3. **Inference (Deriving New Knowledge)**
    - Since $\text{Parent}(John, Alice)$ is true, applying the rule:

$$\text{Child}(Alice, John)$$

    - Similarly, from $\text{Parent}(Mary, Alice)$:

$$\text{Child}(Alice, Mary)$$

Difficult Example: Logic-Based Representation with Quantifiers and Complex Rules

Let's represent a more complex scenario where we have a set of **animals**, and we want to represent the **food chain** and **predator-prey relationships**.

Facts (Knowledge Representation)

1. **Animals and their characteristics:**
   - $\text{Mammal}(Lion)$ → Lion is a mammal.
   - $\text{Mammal}(Elephant)$ → Elephant is a mammal.
   - $\text{Carnivore}(Lion)$ → Lion is a carnivore.
   - $\text{Herbivore}(Elephant)$ → Elephant is a herbivore.
2. **Predator-Prey Relationships:**
   - $\text{Prey}(Deer, Lion)$ → Deer is prey to Lion.
   - $\text{Prey}(Rabbit, Fox)$ → Rabbit is prey to Fox.
   - $\text{Prey}(Elephant, Lion)$ → Elephant is prey to Lion (although it's unlikely in real life, let's assume for this example).

Complex Rule (General Knowledge)

1. **Predator-Prey Relationship Rule:**
   - If **X** is a predator to **Y**, and **Y** is a prey to **Z**, then **Z** is a predator to **X**. This represents the concept that in some ecosystems, the predator-prey dynamics can loop back (circular relationship).

$$\forall X, Y, Z\ (\text{Prey}(Y, X) \land \text{Prey}(Z, Y) \rightarrow \text{Prey}(Z, X))$$

2. **Carnivore-Herbivore Rule:**
   - If **X** is a **carnivore**, and **Y** is a **herbivore**, then **X** can eat **Y**.

$$\forall X, Y\ (\text{Carnivore}(X) \land \text{Herbivore}(Y) \rightarrow \text{Prey}(Y, X))$$

Inference (Deriving New Knowledge)

1. From the **Predator-Prey Relationship Rule**, we know:

$$\text{Prey}(Deer, Lion) \land \text{Prey}(Lion, Elephant) \rightarrow \text{Prey}(Elephant, Deer)$$

   So, based on the circular relationship, Elephant becomes prey to Deer, even though in real life, this is not the case.

2. From the **Carnivore-Herbivore Rule**, since Lion is a carnivore and Elephant is a herbivore, we can infer:

$$\text{Prey}(Elephant, Lion)$$

   (which, although not likely in reality, is inferred by the rule).

The first definition we consider is by Elaine Rich, the author of the book entitled ‘Artificial Intelligence’[1]. It states: Artificial Intelligence is the study of how to make computers do things, at which, at the moment, people are better.

AI is the branch of computer science that deals with symbolic rather than numeric processing and non-algorithmic methods including the rules of thumb or heuristics instead of algorithms as techniques for solving problems.

The definition, by Barr and Feigenbaum in ‘The Handbook of Artificial Intelligence: Artificial Intelligence is the part of computer science concerned with designing intelligent computer systems, i.e., systems that exhibit the characteristics we associate with intelligence in human behaviour.

Symbolic logic may be thought of as a formal language for representing facts about objects and relationships between objects of a problem domain along with a precise inferencing mechanism for reasoning and deduction. An inferencing mechanism derives the knowledge, which is not explicitly/directly available in the knowledge base, but can be logically inferred from what is given in the knowledge base. The reason why the subject matter of the study is called Symbolic Logic is that symbols are used to denote facts about objects of the domain and relationships between these objects. Then the symbolic representations and not the original facts and relationships are manipulated in order to make conclusions or to solve problems.

In the propositional logic, we are interested in declarative sentences, i.e., sentences that can be either true or false, but not both. Any such declarative sentence is called a proposition or a statement. For example

(i) The proposition: “The sun rises in the west,” is False,

(ii) (ii) The proposition: “Sugar is sweet,” is True, and

The truth of the proposition: “Ram has a Ph. D degree.” depends upon whether Ram is actually a Ph. D or not. Though, at present, it may not be known whether the statement is True or False, yet it is sure that the sentence is either True or False and not both True and False simultaneously.

On the other hand, none of the following sentences can be assigned a truth-value, and hence none of these, is a statement or a proposition:

(i) Who was the first Prime Minister of India? (Interrogative sentence)

(ii) Please, give me that book. (Imperative sentence)

(iii) Ram must exercise regularly. (Imperative, rather Deontic)

(iv) Hurrah! We have won the trophy. (Exclamatory sentence)

The symbols, such as P, Q, and R, that are used to denote propositions, are called atomic formulas, or atoms.

P : The sun rises in the west,
Q : Sugar is sweet,
R : Ram has a Ph.D. degree.

We can build, from atoms, more complex propositions, sometimes called compound propositions, by using logical connectives. Examples of such propositions are:

(i) Sun rises in the east and the sky is clear, and

(ii) If it is hot then it shall rain.

The logical connectives in the above two propositions are "and" and "if…then". In the propositional logic, five logical operators or connectives, viz.,

~ (not),

∧ (and),

∨ (or),

→ (if… then), and

↔ (if and only if), are used.

P: The wind speed is high.
Q: Temperature is low.
C: One feels comfortable.

then the sentence:
"If the wind speed is high and the temperature is low, then one does not feel comfortable"
may be represented by the formula
$((P \wedge Q) \rightarrow (\sim C))$.

Logic is the analysis and appraisal of arguments. An argument is a set of statements consisting of a finite number of premises, i.e., assumed statements and a conclusion.
Valid Argument: A valid argument is one in which it would be contradictory for the premises to be true but the conclusion false. In logical studies we are interested in valid arguments.
Example of Valid Argument
(i) If you overslept, you will be late
(ii) You are not late. ∴you did not oversleep.

Example of Invalid argument

(i) If you oversleep, you will be late
(ii) You did not oversleep ∴you are not late (This argument is invalid, because despite not having overslept, one may be late because of some other engagements or lazyness.)

Another Invalid Argument
(i) If we are close to the top of Mt. Everest then we have magnificent view.
(ii) We have a magnificent view. Therefore,
(iii)We are near the top of Mt. Everest.

(This argument is invalid, because, we may have a magnificent view even if we are not close to the top of Mt. Everest. The two given statements do not falsify this claim)

The following argument is valid, but its premises and conclusion both are false:

Premise 1: If moon is made of green cheese Then 2 + 2 = 5
Premise 2: Moon is made of green cheese (False premise)

From Premise 1 and Premise 2, by applying Modus Ponens, we conclude through valid argument that 2 + 2 = 5 (which is False).

However, in order to solve problems of everyday life, we need generally to restrict to only true premises and valid arguments. Then such an argument is called sound argument. Sound Argument: is an argument that is valid and has true premises.

(i) If you are reading this, then you are not illiterate
(ii) (ii) You are reading this (true premise) You are not illiterate (sound conclusion)

Example of valid but not sound argument with correct conclusion.

(i) If moon is made of green cheese then 2 + 2 = 4
(ii) Moon is made of green cheese (False premise)
To conclude 2 + 2 = 4 (correct) makes the argument a Valid Argument

Example of Invalid Argument I
(i) If you overslept, you are late.
(ii) you are late.
Therefore, you overslept.

II If you are in Delhi, you are in India. You are in India. Therefore, you are in Delhi (invalid argument, though the conclusion may be True)

	A	B	~A	$(A \wedge B)$	$(A \vee B)$	$(A \rightarrow B)$	$(A \leftrightarrow B)$
(i)	T	T	F	T	T	T	T
(ii)	T	F	F	F	T	F	F
(iii)	F	T	T	F	T	T	F
(iv)	F	F	T	F	F	T	T

Validity through Truth-Table.

(i) If I overslept, then I am late, i.e., symbolically

$S \rightarrow L$

(ii) I am not late, i.e., symbolically

$\sim L$

To conclude

(iii) I did not oversleep, i.e., symbolically

$\sim S$

To establish the validity/Invalidity of the argument, consider the Truth-Table

S	L	$S \rightarrow L$	$\sim L$	$\sim S$
F	F	T	T	T
F	T	T	F	T
T	F	F	T	F
T	T	T	F	F

There is only one row, viz., first row, in which both the premises viz. $S \rightarrow L$ and $\sim L$ are True. But in this case the conclusion represented by $\sim S$ is also True. Hence, the conclusion is valid.

Invalidity through Truth-Table

(i) If I overslept, then I am late

$S \rightarrow L$

(ii) I did not oversleep, i.e.,

$\sim S$

To conclude

(iii) I would not be late, i.e.,

$\sim L$ (*invalid conclusion*)

S	L	$(S \rightarrow L)$	$\sim S$	$\sim L$
F	F	T	T	T
F	T	T	T	F
T	F	F	F	T
T	T	T	F	F

The invalidity of the argument is established, because, for validity last column must contain True in those rows for which all axioms/premises are True. But in the second row both $S \rightarrow L$ and $\sim S$ are True but $\sim L$ is False

Table 1.6 Truth Table of $(A \wedge B \rightarrow (R \leftrightarrow (\sim S))$

A	B	R	S	~S	$(A \wedge B)$	$(R \leftrightarrow (\sim S))$	$(A \wedge B) \rightarrow (R \leftrightarrow (\sim S))$
T	T	T	T	F	T	F	F
T	T	T	F	T	T	T	T
T	T	F	T	F	T	T	T
T	T	F	F	T	T	F	F
T	F	T	T	F	F	F	T
T	F	T	F	T	F	T	T
T	F	F	T	F	F	T	T
T	F	F	F	T	F	F	T
F	T	T	T	F	F	F	T
F	T	T	F	T	F	T	T
F	T	F	T	F	F	T	T
F	T	F	F	T	F	F	T
F	F	T	T	F	F	F	T
F	F	T	F	T	F	T	T
F	F	F	T	F	F	T	T
F	F	F	F	T	F	F	T

Let us consider the wff G : $(((A \rightarrow B) \wedge A) \rightarrow B)$.

The formula G has 2^2 = 4 possible interpretations in view of the fact it has two atoms viz A and B.

It can be easily seen from the following table that the wff G is True under all its interpretations. Such as a wff which is True under all interpretation is called a valid formula (or a tautology).

Truth Table of $(((A\rightarrow B) \wedge A) \rightarrow B)$

A	B	$(A\rightarrow B)$	$(A\rightarrow B) \wedge A$	$((A \rightarrow B) \wedge A) \rightarrow B$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

Consider another formula

$$G : ((A \rightarrow B) \wedge (A \wedge \sim B))$$

The truth table of the formula G given below shows that G is False under all its interpretations. Such a formula which is False under all interpretations is called an **inconsistent formula (or a contradiction)**.

Truth Table of $(A \rightarrow B) \wedge (A \wedge \sim B)$

A	B	~B	$(A \rightarrow B)$	$(A \wedge \sim B)$	$((A \rightarrow B) \wedge (A \wedge \sim B)$
T	T	F	T	F	F
T	F	T	F	T	F
F	T	F	T	F	F
F	F	T	T	F	F

Definition:

A formula is said to be valid if and only if it is true under all its interpretations.

A formula is said to be invalid if and only if it is not true under at least one interpretation.

A valid formula is also called a Tautology.

A formula is invalid if there is at least one interpretation for which the formula has a truth value False.

From the definitions given above, it is easily seen that

(i) A formula is valid if and only if its negation is inconsistent.

(ii) A formula is invalid if and only if there is at least one interpretation under which the formula is false.

(iii) A formula is consistent if and only if there is at least one interpretation under which the formula is true.

(iv) If a formula is valid, then it is consistent, but not vice versa. (example given below)

(v) If a formula is inconsistent, then it is invalid, but not vice versa. (example given below)

We can verify that the formula **E: ~ (A → B)** is equivalent the formula **G: to A ∧ ~ B** by examining the following truth table. The corresponding values in the last two columns are identical.

Table Joint Truth table of ~ (A → B) and (A ∧ ~ B)

A	B	~ B	(A → B)	~(A → B)	A ∧ ~ B
T	T	F	T	F	F
T	F	T	F	T	T
F	T	F	T	F	F
F	F	T	T	F	F

Table of Equivalences of PL

(1.1)	$E \leftrightarrow G = (E \rightarrow G) \wedge (G \rightarrow E)$	
(1.2)	$E \rightarrow G = \sim E \vee G$	
(1.3)(a)	$E \vee G = G \vee E$;	(b) $E \wedge G = G \wedge E$
(1.4)(a)	$(E \vee G) \vee H = E \vee (G \vee H)$;	(b) $(E \wedge G) \wedge H = E \wedge (G \wedge H)$
(1.5)(a)	$E \vee (G \wedge H) = (E \vee G) \wedge (E \vee H)$;	(b) $E \wedge (G \vee H) = (E \wedge G) \vee (E \wedge H)$
(1.6)(a)	$E \vee \text{False} = E$;	(b) $E \wedge \text{True} = E$
(1.7)(a)	$E \vee \text{True} = \text{True}$	(b) $E \wedge \text{False} = \text{False}$
(1.8)(a)	$E \vee \sim E = \text{True}$;	(b) $E \wedge E = E$
(1.9)	$\sim(\sim E) = E$	
(1.10)(a)	$\sim(E \vee G) = \sim E \wedge \sim G$;	(b) $\sim(E \wedge G) = \sim E \vee \sim G$

2.9 NORMAL FORMS

Some Definitions: A **clause** is a disjunction of literals. For example, $(E \vee \sim F \vee \sim G)$ is a clause. But $(E \vee \sim F \wedge \sim G)$ is not a clause. A **literal** is either an atom, say A, or its negation, say $\sim A$.

Definition: A formula E is said to be in a **Conjunctive Normal Form (CNF)** if and only if E has the form $E : E_1 \wedge \ldots . \wedge E_n$, $n \geq 1$, where each of $E_1, \ldots ., E_n$ is a **disjunction** of literals.

Definition: A formula E is said to be in **Disjunctive Normal Form (DNF)** if and only if E has the form $E: E_1 \vee E_2 \vee \ldots . E_n$, where each E_i is a **conjunction** of literals.

Examples: Let A, B and C be atoms. Then $F: (\sim A \wedge B) \vee (A \wedge \sim B \wedge \sim C)$ is a formula in a disjunctive normal form.

Example: Again G: $(\sim A \vee B) \wedge (A \vee \sim B \vee \sim C)$ is a formula in Conjunctive Normal Form, because it is a conjunction of the two disjunctions of literals viz of $(\sim A \vee B)$ and $(A \vee \sim B \vee \sim C)$

Example: Each of the following is neither in CNF nor in DNF

(i) $(\sim A \vee B) \vee (A \wedge \sim B \vee C)$
(ii) $(A \rightarrow B) \wedge (\sim B \wedge \sim A)$

Using table of equivalent formulas given above, any valid Propositional Logic formula can be transformed into CNF as well as DNF.

The steps for conversion to DNF are as follows

Step 1: Use the equivalences to remove the logical operators '$\leftrightarrow$' and '$\rightarrow$':

(i) $E \leftrightarrow G = (E \rightarrow g) \wedge (G \rightarrow E)$
(ii) $E \rightarrow G = \sim E \vee G$

Step 2 Remove ~'s, if occur consecutively more than once, using

(iii) $\sim (\sim E) = E$

(iv) Use De Morgan's laws to take '~' nearest to atoms

(v) $\sim(E \vee G) = \sim E \wedge \sim G$
(vi) $\sim(E \wedge G) = \sim E \vee \sim G$

Step 3 Use the distributive laws repeatedly

(vii) $E \vee (G \wedge H) = (E \vee G) \wedge (E \vee H)$
(viii) $E \wedge (G \vee H) = (E \wedge G) \vee (E \wedge H)$

Example

Obtain a disjunctive normal form for the formula $\sim(A \rightarrow (\sim B \wedge C))$.

Consider $A \rightarrow (\sim B \wedge C) \;=\; \sim A \vee (\sim B \wedge C)$ *(Using $(E \rightarrow F) = (\sim E \vee F)$)*

Hence, $\sim(A \rightarrow (\sim B \wedge C)) = \sim(\sim A \vee (\sim B \wedge C))$

$= \sim(\sim A) \wedge (\sim(\sim B \wedge C))$ *(Using $\sim(E \vee F) = \sim E \wedge \sim F$)*

$= A \wedge (B \vee (\sim C))$ *(Using $\sim(\sim E) = E$ and $\sim(E \wedge F) = \sim E \vee \sim F$)*

$= (A \wedge B) \vee (A \wedge (\sim C))$ *(Using $E \wedge (F \vee G) = (E \wedge F) \vee (E \wedge G)$)*

However, if we are to obtain CNF of $\sim A\ (\rightarrow (\sim B \wedge C))$, in the last but one step, we obtain

$\sim(A \rightarrow (\sim B \wedge C)) = A \wedge (B \vee \sim C)$, which is in CNF, because, each of A and $(B \vee \sim C)$ is a disjunct.

Example: Obtain conjunctive Normal Form (CNF) for the formula: $D \rightarrow (A \rightarrow (B \wedge C))$

Consider

$D \rightarrow (A \rightarrow (B \wedge C))$ *(using $E \rightarrow F = \sim E \vee F$ for the inner implication)*

$= D \rightarrow (\sim A \vee (B \wedge C))$ *(using $E \rightarrow F = \sim E \vee F$ for the outer implication)*

$= \sim D \vee (\sim A \vee (B \wedge C))$

$= (\sim D \vee \sim A) \vee (B \wedge C)$ *(using Associative law for disjunction)*

$= ((\sim D \vee \sim A \vee B) \wedge (\sim D \vee \sim A \vee C)$

The last line denotes the conjunctive Normal Form of $D \rightarrow (A \rightarrow (B \wedge C))$ (using distributivity of $\vee$ over $\wedge$)

Note: *If we stop at the last but one stop, then we obtain $(\sim D \vee \sim A) \vee (B \wedge C) = \sim D \vee \sim A \vee (B \wedge C)$ is a* ***Disjunctive Normal Form*** *for the given formula: $D \rightarrow (A \rightarrow (B \wedge C))$*

2.10 LOGICAL DEDUCTION

Definition: A formula G is said to be a logical **consequence** of given formulas $E_1,\ldots,E_n$ **(or G is logical derivation of $E_1,\ldots.E_2$)** if and only for any interpretation I in which $E_1 \wedge E_2 \wedge \ldots \wedge E_n$ is true, for the interpretation I, G is also true. The proposition $E_1, E_2\ldots,E_n$ are called *axioms/premises* of G.

Next, we state without proof two very useful theorems for establishing logical derivations:

Theorem 1: Given formulas $E_1,\ldots, E_n$ and a formula G, G is a logical derivation of $E_1,\ldots.,E_n$ if and only if the formula $((E_1 \wedge \ldots.\wedge E_n) \rightarrow G)$ is valid, i.e., True for all interpretations of the formula.

Theorem 2: Given formulas $E_1,\ldots,E_n$ and a formula G, G is a logical consequence or derivation of $E_1,\ldots E_n$ if only if the formula $(E_1 \wedge \ldots.\wedge E_n \wedge \sim G)$ is inconsistent, i.e., False for all interpretations of the formula.

Example: We are given the formulas

$E_1 : (A \rightarrow B)$, $E_2 : \sim B$, $G : \sim A$

We are required to show that G is a logical consequence of E_1 and E_2.

Method 1: From the following Table, it is clear that whenever E_1: $A \rightarrow B$ and E_2: $\sim B$ both are simultaneously True, (*which is true only in the last row of the table*) then G: $\sim A$ is also True. Hence, the proof.

A	B	A→B	~ B	~ A
T	T	T	F	F
T	F	F	T	F
F	T	T	F	T
F	F	T	T	T

Method 2: We prove the result by showing the validity of $E_1 \wedge E_2 \rightarrow G$, i.e., of $((A \rightarrow B) \wedge \sim B) \rightarrow \sim A$ by transforming it into a conjunctive normal form.

$$(A \rightarrow B) \wedge \sim B) \rightarrow \sim A = \sim ((A \rightarrow B) \wedge \sim B) \vee \sim A \quad (\text{using } E \rightarrow F = (\sim E \vee F))$$
$$= \sim ((\sim A \vee B) \wedge \sim B) \vee \sim A$$

$$= \sim ((\sim A \wedge \sim B) \vee (B \wedge \sim B)) \vee \sim A$$
$$= \sim ((\sim A \wedge \sim B) \vee \text{False}) \vee \sim A$$
$$= \sim ((\sim A \wedge \sim B)) \vee \sim A \quad (\textit{using De Morgan's Laws})$$
$$= (A \vee B) \vee \sim A =$$
$$= (B \vee A) \vee \sim A$$
$$= B \vee (A \vee \sim A)$$
$$= B \vee \text{True}$$

In addition to the baggage of concepts of propositional logic, FOPL has the following additional concepts: terms, predicates and quantifiers. These concepts will be introduced at appropriate places.

In order to have a glimpse at how FOPL extends propositional logic, let us again discuss the earlier argument.

Every man is mortal. Raman is a man.
Hence, he is mortal.

In order to derive the validity of above simple argument, in stead of looking at an atomic statement as indivisible, to begin with, we divide each statement into *subject* and *predicate*. The two predicates which occur in the above argument are:

'*is mortal*' and '*is man*'.

Let us use the notation
IL: *is_mortal* and
IN: *is_man*.

In view of the notation, the argument on para-phrasing becomes:
For all x, if IN (x) then IL (x).
IN (Raman).
Hence, IL (RAMAN)

For this purpose, from the discussion in the Introduction, we need at least the following concepts.

i) ***Individual symbols or** constant symbols*: These are usually names of objects, such as Ram, Mohan, numbers like 3, 5 etc.

ii) **Variable symbols**: These are usually lowercase unsubscripted or subscripted letters, like x, y, z, x_3.

iii) **Function symbols:** These are usually lowercase letters like f, g, h,….or strings of lowercase letters such as *father* and *product*.

iv) **Predicate symbols**: These are usually uppercase letters like P, Q, R,….or strings of lowercase letters such as *greater-than*, *is_tall* etc.

In order to symbolize the following statements:

i) There exists a number that is rational.
ii) Every rational number is a real number
iii) For every number *x*, there exists a number *y*, which is greater than *x*.

let us denote *x is a rational number* by Q(x), *x is a real number* by R(x), and *x is less than y* by *LESS(x, y)*. Then the above statements may be symbolized respectively, as

(i) $(\exists x)\ Q(x)$
(ii) $(\forall x)\ (Q(x) \rightarrow R(x))$
(iii) $(\forall x)\ (\exists y)\ LESS(x, y)$.

Example

Translate the statement: *Every man is mortal. Raman is a man. Therefore, Raman is mortal.*

As discussed earlier, let us denote "*x is a man*" by MAN (x), and "*x is mortal*" by MORTAL(x). Then "*every man is mortal*" can be represented by

$$(\forall x)\ (\text{MAN}(x) \rightarrow \text{MORTAL}(x)),$$

"Raman is a man" by

$$\text{MORTAL}(\text{Raman}).$$

The whole argument can now be represented by

$$(\forall x)\ (\text{MAN}(x) \rightarrow \text{MORTAL}(x)) \wedge \text{MAN (Roman)} \rightarrow \text{MORTAL (Roman)}.$$

as a single statement.

In order to further explain symbolisation let us recall the axioms of natural numbers:

Next we discuss some equivalences, and inequalities

The following equivalences hold for any two formulas P(x) and Q(x):

(i) $(\forall x)\, P(x) \wedge (\forall x)\, Q(x) = (\forall x)\, (P(x) \wedge Q(x))$

(ii) $(\exists x)\, P(x) \vee (\exists x)\, Q\, (x) = (\exists x)\, (P(x) \vee Q(x)$

But the following inequalities hold, in general:

(iii) $(\forall x)\, (P(x) \vee Q(x) \neq (\forall x)\, P(x) \vee (\forall x)\, Q(x)$

(iv) $(\exists x)\, (P(x) \wedge Q(x) \neq (\exists x)\, P(x) \wedge (\exists x)\, Q\, (x)$

Equivalences involving Negation of Quantifiers

(v) $\sim (\forall x)\, P(x) = (\exists x) \sim P(x)$

(iv) $\sim (\exists x)\, P(x) = (\forall x) \sim P(x)$

Examples: For each of the following closed formula, Prove

(i) $(\forall x)\, P(x) \wedge (\exists y) \sim P(y)$ is inconsistent.

(ii) $(\forall x)\, P(x) \rightarrow (\exists y)\, P(y)$ is valid

Solution: (i) Consider

$$(\forall x)\, P(x) \wedge (\exists y) \sim P(y)$$

$$= (\forall x)\, P(x) \wedge \sim (\forall y)\, P(y) \quad (\textit{taking negation out})$$

But we know for each bound occurrence, a variable is dummy, and can be replaced in the whole scope of the variable uniformly by another free variable. Hence,

$R = (\forall x)\, P(x) \wedge \sim (\forall x)\, P(x)$

Each conjunct of the formula is either

True of False and, hence, can be thought of as a formula of PL, in stead of formula of FOPL, Let us replace $(\forall x)$ $(P(x)$ by Q , a formula of PL.

$R = Q \wedge \sim Q =$ False

Hence, the proof.

(ii) Consider

$(\forall x)\ P(x) \rightarrow (\exists y)\ P(y)$

Replacing '$\rightarrow$' we get

$= \sim (\forall x)\ P(x) \vee (\exists y)\ P(y)$

$= (\exists x) \sim P(x) \vee (\exists y)\ P(y)$

$= (\exists x) \sim P(x) \vee (\exists x)\ P(x)$ *(renaming x as y in the second disjunct)*

In other words,

$= (\exists x)\ (\sim P(x) \vee P(x))$ *(using equivalence)*

The last formula states: *there is at least one element say b, for $\sim P(b) \vee P(b)$ holds i.e., for b, either P(b) is False or P(b) is True.*

But, as P is a predicate symbol and b is a constant $\sim P(b) \vee P(b)$ must be True. Hence, the proof.

1.3 PRENEX NORMAL FORM

In order to facilitate problem solving through PL, we discussed two normal forms, viz, the conjunctive normal form **CNF** and the disjunctive normal form **DNF**. In **FOPL**, there is a normal form called the **prenex normal form**. The use of a prenex normal form of a formula simplifies the proof procedures, to be discussed.

Definition A formula G in FOPL is said to be in a **prenex normal form** if and only if the formula G is in the form

$$(Q_1x_1)\ldots.(Q_n\ x_n)\ P$$

where each (Q_ix_i), for i = 1, ….,n, is either $(\forall x_i)$ or $(\exists x_i)$, and P is a quantifier free formula. The expression $(Q_1x_1)\ldots.(Q_n\ x_n)$ is called the **prefix** and P is called the **matrix of the formula G**.

Examples of some formulas in prenex normal form:

(i) $(\exists x)\ (\forall y)\ (R(x, y) \vee Q(y)), (\forall x)\ (\forall y)\ (\sim P(x, y) \rightarrow S(y))$,

(ii) $(\forall x)\ (\forall y)\ (\exists z)\ (P(x, y) \rightarrow R\ (z))$.

Then, the following laws involving quantifiers hold good in FOPL

(i) $(Qx)\,P[x] \vee G = (Qx)\,(P[x] \vee G)$.

(ii) $(Qx)\,P[x] \wedge G = (Qx)\,(P[x] \wedge G)$.

In the above two formulas, Q may be either $\forall$ or $\exists$.

(iii) $\sim((\forall x)\,P[x]) = (\exists x)\,(\sim P[x])$.

(iv) $\sim((\exists x)\,P[x]) = (\forall x)\,(\sim P[x])$.

(v) $(\forall x)\,P[x] \wedge (\forall x)\,H[x] = (\forall x)\,(P[x] \wedge H[x])$.

(vi) $(\exists x)\,P[x] \vee (\exists x)\,H[x] = (\exists x)\,(P[x] \vee H[x])$.

That is, the universal quantifier $\forall$ and the existential quantifier $\exists$ can be distributed respectively over $\wedge$ and $\vee$.

But we must be careful about *(we have already mentioned these inequalities)*

(vii) $(\forall x)\,E[x] \vee (\forall x)\,H[x] \neq (\forall x)\,(P[x] \vee H[x])$ and

(viii) $(\exists x)\,P[x] \wedge (\exists x)\,H[x] \neq (\exists x)\,(P[x] \wedge H[x])$

Steps for Transforming an FOPL Formula into Prenex Normal Form

Step 1 Remove the connectives '↔' and '→' using the equivalences

$$P \leftrightarrow G = (P \rightarrow G) \wedge (G \rightarrow P)$$
$$P \rightarrow G = \sim P \vee G$$

Step 2 Use the equivalence to remove even number of ~'s

$$\sim(\sim P) = P$$

Step 3 Apply De Morgan's laws in order to bring the negation signs immediately before atoms.

$$\sim(P \vee G) = \sim P \wedge \sim G$$
$$\sim(P \wedge G) = \sim P \vee \sim G$$

and the quantification laws

$$\sim((\forall x)\ P[x]) = (\exists x)\ (\sim P[x])$$
$$\sim((\exists x)\ P[x]) = (\forall x)\ (\sim F[x])$$

Step 4 rename bound variables **if necessary**

Step 5 Bring quantifiers to the left before any predicate symbol appears in the formula. This is achieved by using (i) to (vi) discussed above.

$$(Q1\ x)\ P[x] \vee (Q2\ x)\ H[x] = (Q1\ x)\ (Q2\ z)\ (P[x] \vee H[z])$$
$$(Q3\ x)\ P[x] \wedge (Q4\ x)\ H[x] = (Q3\ x)\ (Q4\ z)\ (P[x] \wedge H[z])$$

Example: Transform the following formulas into prenex normal forms:

(i) $(\forall x)\,(Q(x) \rightarrow (\exists x)\,R\,(x, y))$
(ii) $(\exists x)\,(\sim (\exists y)\,Q(x, y) \rightarrow ((\exists z)\,R(z) \rightarrow S\,(x)))$
(iii) $(\forall x)\,(\forall y)\,((\exists z)\,Q(z, y, z) \wedge ((\exists u)\,R\,(x, u) \rightarrow (\exists v)\,R\,(y, v)))$.

Part (i)

Step 1: By removing '→', we get

$$(\forall x)\,(\sim Q\,(x) \vee (\exists x)\,R\,(x, y))$$

Step 2: By renaming x as z in (∃x) R (x, y) the formula becomes

$$(\forall x)\,(\sim Q\,(x) \vee (\exists z)\,R\,(z, y))$$

Step 3: As ~ Q(x) does not involve z, we get

$$(\forall x)\,(\exists z)\,(\sim Q\,(x) \vee R\,(z, y))$$

Part (ii)

$(\exists x)\,(\sim (\exists y)\,Q\,(x, y) \rightarrow ((\exists z)\,R\,(z) \rightarrow S\,(x)))$

Step 1: Removing outer '→' we get

$$(\exists x)\,(\sim (\sim ((\exists y)\,Q\,(x, y))) \vee ((\exists z)\,R\,(z) \rightarrow S\,(x)))$$

Step 2: Removing inner '→', and simplifying ~ (~ ( )) we get

$$(\exists x)\,((\exists y)\,Q\,(x, y) \vee (\sim ((\exists z)\,R(z)) \vee S\,(x)))$$

Step 3: Taking '~' inner most, we get

$$(\exists x)\,(\exists y)\,Q\,(x, y) \vee ((\forall z) \sim R(z) \vee S(x)))$$

As first component formula Q (x, y) does not involve z and S(x) does not involve both y and z and ~ R(z) does not involve y. Therefore, we may take out (∃ y) and (∀z) so that, we get

$(\exists x)\,(\exists y)\,(\forall z)\,(Q\,(x, y) \vee (\sim R(z) \vee S\,(x)\,)$, which is the required formula in prenex normal form.

Part (iii)

$(\forall x)\ (\forall y)\ ((\exists z)\ Q\ (x, y, z) \wedge ((\exists u)\ R\ (x, u) \rightarrow (\exists v)\ R\ (y\ v)))$

Step 1: Removing '$\rightarrow$', we get

$$(\forall x)\ (\forall y)\ ((\exists z)\ Q\ (x, y, z) \wedge (\sim ((\exists u)\ R\ (x, u)) \vee (\exists v)\ R\ (y, v)))$$

Step 2: Taking '$\sim$' inner most, we get

$$(\forall x)\ (\forall y)\ ((\exists z)\ Q\ (x, y, z) \wedge ((\forall u) \sim R\ (x, u) \vee (\exists v)\ R\ (y, v)))$$

Step 3: As variables z, u & v do not occur in the rest of the formula except the formula which is in its scope, therefore, we can take all quantifiers outside, preserving the order of their occurrences, Thus we get

$$(\forall x)\ (\forall y)\ (\exists z)\ (\forall u)\ (\exists v)\ (Q\ (x, y, z) \wedge (\sim R\ (x, u) \vee R\ (y, v)))$$

1.4 (SKOLEM) STANDARD FORM

A further refinement of Prenex Normal Form (PNF) called (Skolem) Standard Form, is the basis of problem solving through Resolution Method. The Resolution Method will be discussed in the next unit of the block.

The **Standard Form of a formula of FOPL is** obtained through the following three steps:

(1) The given formula should be converted to Prenex Normal Form (PNF), and then
(2) Convert the Matrix of the PNF, i.e, quantifier-free part of the PNF into conjunctive normal form
(3) Skolomization: Eliminate the existential quantifiers using skolem constants and functions

Before illustrating the process of conversion of a formula of FOPL to Standard Normal Form, through examples, we discuss briefly skolem functions.

Skolem Function

We in general, mentioned earlier that $(\exists x)\ (\forall y)\ P(x,y) \neq\ (\forall y)\ (\exists x)\ P(x,y)$…….(1)

For example, if P(x,y) stands for the relation 'x>y' in the set of integers, then the L.H.S. of the inequality (i) above states: *some (fixed) integer (x) is greater than all integers (y)*. This statement is False.

On the other hand, R.H.S. of the inequality (1) states: *for each integer y, there is an integer x so that x>y*. This statement is True.

The difference in meaning of the two sides of the inequality arises because of the fact that on L.H.S. *x* in $(\exists x)$ is independent of *y* in $(\forall y)$ **whereas** on R.H.S *x* of dependent on *y*. In other words, *x* on L.H.S. of the inequality can be replaced by some constant say 'c' whereas on the right hand side *x* is some function, say, f(y) of *y*.

Therefore, the two parts of the inequality (i) above may be written as
LH.S. of (1) $= (\exists x)\ (\forall y)\ P\ (x,y) = (\forall y)\ P(c,y)$,

Dropping x because there is no x appearing in $(\forall y)\ P(c,y)$

R.H.S. of (1) = $(\forall y)\ (\exists x)\ P(f(y),y) = (\forall y)\ P(f(y), y)$

The above argument, in essence, explains what is meant by each of the terms viz. *skolem constant, skolem function and skolomisation.*

The constants and functions which replace existential quantifiers are respectively called **skolem constants and skolem functions.** The process of replacing all existential variables by skolem constants and variables is called **skolemisation**.

A form of a formula which is obtained after applying the steps for

(i) reduction to PNF and then to
(ii) CNF and then
(iii) applying skolomization is called **Skolem Standard Form** or just **Standard Form.**

We explain through examples, the skolomisation process after PNF and CNF have already been obtained.

Example: Skolomize the following:

(i) $(\exists x_1)\ (\exists x_2)\ (\forall y_1)\ (\forall y_2)(\exists x_3)(\forall y_3)\ P(x_1, x_2, x_3, y_1, y_2, y_3)$

(ii) $(\exists x_1)(\forall y_1)(\exists x_2)(\forall y_2)\ (\exists x_3)P(x_1, x_2, x_3, y_1, y_2) \wedge (\exists x_1)(\forall y_3)(\exists x_2)\ (\forall y_4)Q(x_1, x_2, y_3, y_4)$

Solution (i) As existential quantifiers x_1 and x_2 precede all universal quantifiers, therefore, x_1 and x_2 are to be replaced by *constants*, but by distinct constants, say by 'c' and 'd' respectively. As existential variable x_3 is preceded by universal quantifiers y_1 and y_2, therefore, x_3 is replaced by some function $f(y_1, y_2)$ of the variables y_1 and y_2 After making these substitutions and dropping universal and existential variables, we get the skolemized form of the given formula as
$(\forall y_1)\ (\forall y_2)\ (\forall y_3)\ (c, d, f\ (y_1, y_2), y_1, y_2, y_3)$.

Solution (ii) As a first step we must bring all the quantifications in the beginning of the formula through Prenex Normal Form reduction. Also,

$$(\exists x)\ldots P(x,\ldots) \wedge (\exists x)\ldots.\ Q\ (x,\ldots.) \neq (\exists x)\ (\ldots.P(x) \wedge \ldots Q(x,\ldots.),$$

therefore, we rename the second occurrences of quantifiers $(\forall x_1)$ and $(\forall x_2)$ by renaming these as x_5 and x_6. Hence, after renaming and pulling out all the quantifications to the left, we get

$$(\exists x_1)\ (\forall y_1)\ (\exists x_2)\ (\forall y_2)\ (\exists x_3)\ (\exists x_5)\ (\forall y_3)\ (\exists x_6)\ (\forall y_4)$$
$$(P(x_1, x_2, x_3, y_1, y_2) \wedge Q\ (x_5, x_6, y_3, y_4)$$

AI and ML Unit 2

IGNOU Notes

AI and ML Unit 2

IGNOU Notes

Dr Raj Kumar Sharma

profrksharma07@gmail.com

9818469257