SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Artificial Intelligence and Data Science

Course Name – Introduction to Artificial Intelligence

II Year / III Semester

Unit 2 Predicate Logic

Predicate Logic

Predicate: It tells something about the subject. sentence involving the predicates that describe the property of objects are denoted by P(x) where
P – The Predicate
x – It is a variable denoting any object.

Characteristics of Predicate Logic:
1. Logical inferencing is allowed.
2. More accurate KR of facts of the real world.
3. Program designing is its application area.
4. Better theoretical foundation.
5. A predicate with no variable is called a Ground Atom.

Constants
– Names of specific objects – E.g. doreen, gord, william, 32
• Functions
– Map objects to objects – E.g. father(doreen), age(gord), max(23,44)
• Variables
– For statements about unidentified objects or general statements – E.g.x,y,z,...

Terms represent objects
• The set of terms is inductively defined by the following rules:
– Constants:Anyconstantisaterm
– Variables:Anyvariableisaterm
– Functions:Any Expressions(t1,...,tn)of arguments(where each argument is a term and f is a function of arity n) is a term
• Terms without variables are called ground terms
• Examples:
–c
– f(c)
– g(x,x)
– g(f(c), g(x,x))

Rules of Predicate calculus symbols:
1. Set of letters (uppercase or lowercase) is allowed.
2. Set of digits (0-9) is allowed.
3. Underscore (_) is allowed.
4. Blanks and non-alphanumeric characters can't be used.
5. Special characters like $, *, #, and / are not allowed.

Symbols:
1. Predicate Symbols: It denotes relations or functional mapping from the elements of a domain to the values true or false.

2. Function Symbols: It denotes relations defined on a domain.

3. Variable Symbols: Lowercase unsubscribed or subscribed letters like x, y, z, t, u, v, etc. It can assume different values over a given domain.

4. Constants: It is fixed value term. They are individual symbols which are names of objects like 1, 2, 3, etc.

5. Quantifiers: There are two types of Quantifiers:
i. Existential Quantifier (∃): It means for some x or there is no x.
ii. Universal Quantifier (∀): It means for all x.

6. Logical Operators: There are five type of logical operators such as – not(~), and (^), or(v), implication, equivalence.

Two quantifiers: Universal (∀) and Existential (∃)

• Allow us to express properties of collections of objects instead of enumerating objects by name

– Apply to sentence containing variable • Universal ∀: true for all substitutions for the variable

– “for all”: ∀<variables> <sentence> • Existential ∃: true for at least one substitution for the variable

– “there exists”: ∃<variables> <sentence>

• Examples:

– ∃ x: Mother(art) = x

– ∀ x ∀ y: Mother(x) = Mother(y) → Sibling(x,y)

– ∃ y ∃ x: Mother(y) = x

The set of formulas is inductively defined by the following rules:

1. Preciate symbols: If P is an n-ary predicate symbol and t1,...,tn are terms then P(t1,...,tn) is a formula.

2. Negation: If ϕ is a formula, then ¬ϕ is a formula

3. Binary connectives: If ϕ and ψ are formulas, then (ϕ → ψ) is a formula. Same for other binary logical connectives.

4. Quantifiers: If ϕ is a formula and x is a variable, then ∀xϕ and ∃xϕ are formulas.

• Atomic formulas are formulas obtained only using the first rule

Any occurrence of a variable in a formula not in the scope of a quantifier is said to be a free occurrence

- Otherwise it is called a bound occurrence
- Thus,ifxisafreevariableinφ,itisboundin∀xφ and
∃xφ
- A formula with no free variables is called a closed
formula
- Example: x and y are bound variables, z is a free variable

$\forall x \forall y (P(f(x)) \rightarrow \neg(P(x)) \rightarrow Q(f(y), x, z)))$

S := <Sentence> <Sentence> :=
<AtomicSentence> | <Sentence> <Connective> <Sentence> | <Quantifier> <Variable>,... <Sentence>
| ¬ <Sentence> | (<Sentence>)
<AtomicSentence> := <Predicate> (<Term>, ...) <Term> := <Function> (<Term>, ...)
| <Constant>
| <Variable> <Connective>:= ∧|v|→|<->
<Quantifier> := |
<Constant> := “c" | “x1" | “john" | ...
<Variable> := "a" | "x" | "s" | ...
<Predicate> := “before" | “hasColor" | “raining" | ... <Function> := “mother" | “leftLegOf" | ...