innovate achieve lead

BITS Pilani
Pilani Campus

# CS/IS F214
# Logic in Computer Science

## MODULE: PREDICATE LOGIC

**Expressing using Predicates : Examples**

**Experessing using Predicates: Horn-Clause Form and Prolog**

# Specifications using Predicate Logic

- Examples:
  - *No student attended every lecture*
    - **$\forall$X $\neg$ ($\forall$Y student(X) $\wedge$ lecture(Y) --> attended(X,Y))**
      - Exercises:
        - Rewrite this (*without changing the meaning*)
          1. by replacing the outermost quantifier with an existential quantifier.
          2. by replacing the innermost quantifier with an existential quantifier.
          3. such that the scope of the inner quantifier is the smallest

# Specifications using Predicate Logic

- Examples:
  i. *No student attended every lecture*
     - $\forall X \neg (\forall Y \text{ student}(X) \wedge \text{lecture}(Y) \dashrightarrow \text{attended}(X,Y))$
  ii. *No lecture was attended by every student*
     - $\forall Y \neg (\forall X \text{ student}(X) \wedge \text{lecture}(Y) \dashrightarrow \text{attended}(X,Y))$
     - Exercises:
       i. Does (i) imply (ii)?
       ii. Does (ii) imply (i)?
       iii. If you swap variables X and Y in the quantifier positions will the formula in (ii) remain the same?
       iv. If you swap all occurrences of variables X and Y will the formula in (ii) remain the same?

# Exercises (from the text book: Huth & Ryan).

- Write the following in Predicate Logic
  i. *All red things are in the box*
     - **$\forall$T red(T) --> inBox(T)**
  ii. *Only red things are in the box*
     - **$\forall$T inBox(T) --> red(T)**

# Exercises (adapted from the text book: Huth & Ryan).

- Write the following in Predicate Logic
  - *Raj is Jaya's cousin*
- Given a ***cousin*** relation defined, this would do:
  - **cousin("Raj","Jaya")**
- Otherwise one has to define a cousin relation.

# Exercises (adapted from the text book: Huth & Ryan).

- Define a ***cousin*** property using ***father***, ***mother***, ***sister***, ***brother***:
  - **$\forall$Any1 $\forall$C cousin(Any1,C) <--**
    **$\exists$Pa $\exists$Pc parent(Any1,Pa) $\wedge$ parent(C,Pc) $\wedge$ sibling(Pa,Pc)**
  - **$\forall$A $\forall$B sibling(A,B) <-- brother(A,B) $\vee$ sister(A,B)**
  - **$\forall$Any1 $\forall$Pa parent(Any1,Pa) <-- mother(A,Pa) $\vee$ father(A,Pa)**

# Exercises (from the text book: Huth & Ryan).

- Write the following in Predicate Logic
  - All brothers are siblings
- Given a predicate **brother(X,Y)** and **sibling(X,Y)** to indicate X's brother is Y and X's sibling is Y,
  - **∀X ∀Y brother(X,Y) --> sibling(X,Y).**

# Exercises (from the text book: Huth & Ryan).

- Write the following in Predicate Logic
  i. An attacker can persuade a server that a successful login has occurred even if it hasn't.
- Given the predicates
  - **attacker(X)** // X is an attacker
  - **server(S)** // S is a server
  - **persuade(A, X, Y)** // A can persuade X about Y
  - **login(S, token(S, T))**
    - // attempt to login into S results in token(S,T) where T is TRUE or FALSE
- the statement above can be encoded as:
  - **∀S ∀A attacker(A) ∧ server(S) -->**
    **(∀T login(S,token(S,T)) --> persuade(A,S,token(S,TRUE)))**

# Encoding in Predicate Logic: Example

- Axioms of Group (G,+):
    - Closure:
    - Associativity:
    - Existence of Identity
    - Existence of Inverse

# Encoding in Predicate Logic: Example

- Definition of natural numbers (*incomplete*):
  - **$\forall$X (equals(X,0) $\vee$ $\exists$Y equals(X, succ(Y)) --> natural(X)**

- **Note**: *We must insist Y to be a natural number* – this will require a recursive definition. **End of Note**
  - Exercise: *Write such a recursive definition.*

# Prolog Programming and Horn Clauses

- Prolog uses Horn Clauses as the basis for programming:
  - A Horn Clause is an implication with zero or more ***antecedents*** and one ***implicand***:
    - All antecedents and the implicand are predicates.
  - i.e. a Horn Clause is of the form:
    - $p_1(T_{11},\ldots,T_{1K1}) \wedge p_2(T_{21},\ldots,T_{2K2}) \wedge \ldots \wedge p_m(T_{m1},\ldots,T_{mKm}) \text{-->} q(T_{q1},\ldots,T_{qKq})$
      - where each $p_i$ is a predicate name and each $T_{ij}$ is a term: i.e.
        - a constant
        - a variable or
        - a function term
- A single predicate $p(T_1,\ldots,T_K)$ is a degenerate implication:
  - TRUE --> $p(T_1,\ldots,T_K)$

# Prolog Programming and Horn Clauses

- In Prolog, a typical Horn Clause of the form
  - $p_1(T_{11},\ldots,T_{1K1}) \wedge p_2(T_{21},\ldots,T_{2K2}) \wedge \ldots \wedge p_m(T_{m1},\ldots,T_{mKm})\text{-->}q(T_{q1}\ldots,T_{qKq})$
- is represented as:
  - $q(T_{q1}\ldots,T_{qKq})$ :- $p_1(T_{11},\ldots,T_{1K1}), p_2(T_{21},\ldots,T_{2K2}), \ldots, p_m(T_{m1},\ldots,T_{mKm})$.

    i.e.

    - the implicand is on the left most end,
    - the antecedents occur on the right of :- (read this as <--),
    - the commas separating the antecedents indicate conjunction, and
    - there is a period ending the clause.

# Prolog Programming and Horn Clauses

- A Prolog program is a conjunction of Horn Clauses and the conjunction is implicit:
  - i.e. syntactically, a Prolog program is a list of Horn Clauses.
- A degenerate clause (with a single predicate) is referred to as a ***fact*** in Prolog: e.g.
  - nat(0).
- A typical Horn Clause is referred to as a ***rule*** in Prolog: e.g.
  - nat(s(X)):-nat(X).
    - Note: A fact is a special form of a rule. End of Note.
- Argue that these two rules form a specification of *natural numbers* in Prolog.
- Exercise:
  - Specify the *addition* operation in Prolog.