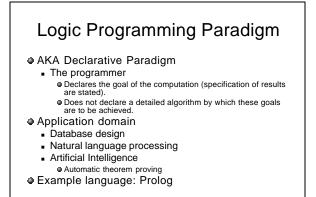


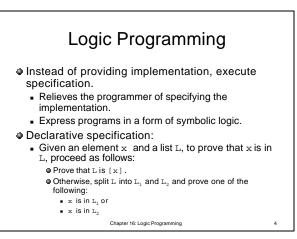
## Topics

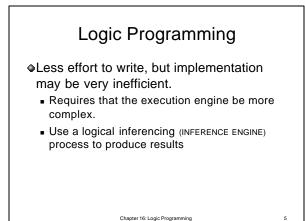
Chapter 16: Logic Programming

Introduction
Predicate Calculus
Propositions
Clausal Form
Horn Clauses



Chapter 16: Logic Programming





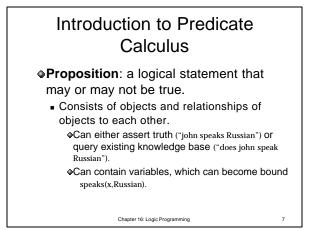
#### Introduction to Predicate Calculus

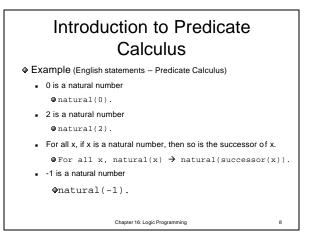
 Symbolic logic can be used for the basic needs of formal logic:

- Express propositions
- Express relationships between propositions
- Describe how new propositions can be inferred from other propositions

 Particular form of symbolic logic used for logic programming is called *first-order* predicate calculus

```
Chapter 16: Logic Programming
```





### Introduction to Predicate Calculus

- First and third logical statements are axioms for the natural numbers.
  - Statements that are assumed to be true and from which all true statements about natural numbers can be proved.
- Second logical statement can be proved from the previous axioms.
  - 2 = successor(successor(0)).
  - natural(0) → natural(successor(successor(0)).
- Fourth logical statement cannot be proved from the axioms and so can be assumed to be false. Chapter 16: Logic Programming 9

# Predicate Calculus: statements

- Predicate calculus classifies the different parts of statements as:
  - *Constants*. These are usually number or names. Sometimes they are called *atoms*, since they cannot be broken down into subparts.
     Example: 1, 0, true, false
  - Predicates. These are names for functions that are true or false, like Boolean functions in a program.
    - Can take any number of arguments.
    - Example: the predicate natural takes one argument.

10

Chapter 16: Logic Programming

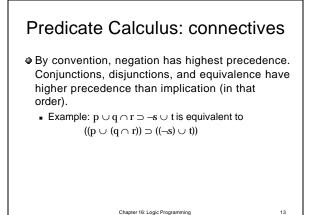
# Functions. Predicate calculus distinguishes between functions that are true or false – these are predicates – and all other functions, which represent non-Boolean values. Example: successor Variables. Variables stand for as yet unspecified quantities. Example: x Connectives. These include the operations and, or , and not, just like the operations on Boolean data in programming languages. Additional connectives are implication and equivalence

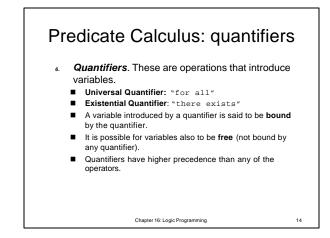
Chapter 16: Logic Programming

11

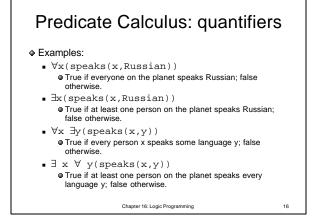
# Predicate Calculus: table of

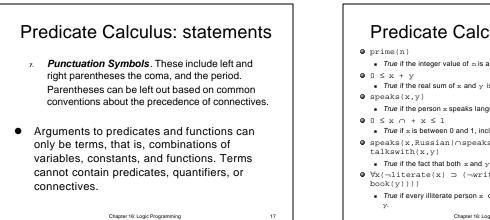
| Name       | \$ymbol | Example | ECTIVES<br>Meaning         | Notes                                    |
|------------|---------|---------|----------------------------|--|
| legation   | ~       | -a      | not a                      | True if a is false;<br>athorwise false   |
| onjunction | 0       | a∩b     | a and b                    | True if a and b are both true            |
| isjunction | U       | a∪b     | aorb                       | True if either a or b                    |
| quivalence | =       | a≡ b    | a is equivalent to b       | True if a and b are<br>both true or both |
| nplication | ⊃       |         | a implies b<br>b implies a | false<br>Logically equivalent            |

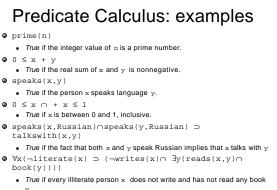




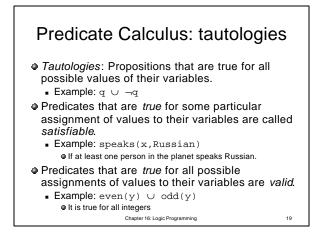
| Predicate Calculus: table of<br>quantifiers |        |         |   |   |  |
|---|--------|---------|---|---|--|
| Name  | Symbol | Example | Meaning                                       | 1 |  |
| Universal                                   | A      | ∀хр     | For all X, P is true                          |   |  |
| Existential                                 | Э      | Эхр     | There exists a value of x such that ℙ is true |   |  |
|   |        | •       |   | - |  |
| Chapter 16: Logic Programming 15            |        |         |   |   |  |

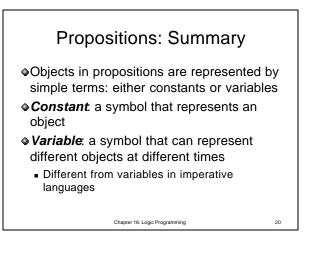


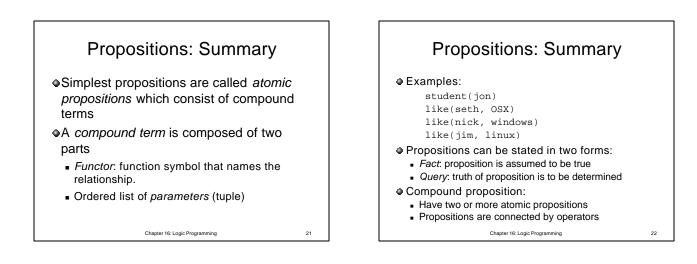


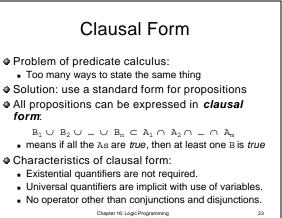


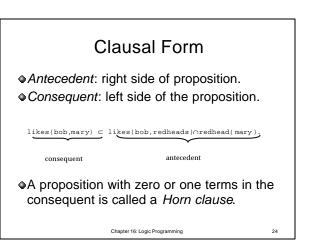
Chapter 16: Logic Programming

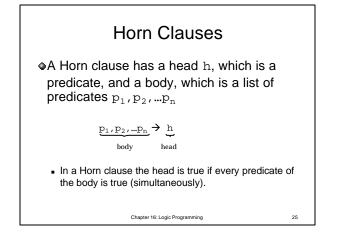


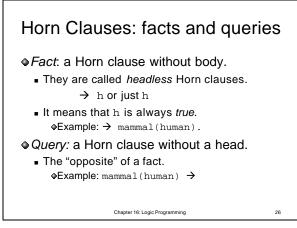












| From Predicates to Horn<br>Clauses   |
|--|
| <ul> <li>There is a limited correspondence between<br/>Horn clauses and predicates.</li> </ul> |
| <ul> <li>Horn clauses can be written equivalently as a<br/>predicate</li> </ul>                |
| $\oplus HC$ : snowing(C) $\leftarrow$ precipitation(C), freezing(C).                           |
| <b>♦PC:</b> snowing(C) ⊂ precipitation(C) $\cap$ freezing(C).                                  |
| <ul> <li>Not all predicates can be translated into Horn<br/>clauses.</li> </ul>                |

Chapter 16: Logic Programming

**Properties of Predicate Logic** Expressions Meaning roperty mmutativity  $\vee q \equiv q \vee p$  $q \equiv q \land p$  $\lor q) \lor r \equiv p \lor (q \lor r)$  $\land q) \land r \equiv p \land (q \land r)$ sociativity stributivity  $\lor q \land r \equiv (p \lor q) \land (p \lor r)$  $\wedge (q \lor r) \equiv (p \land q) \lor (p \land r)$ / n = n  $\mathbf{n} = \mathbf{n}$ entity √ ¬p ≡ true ¬p ≡ false eMorgan  $(p \lor q) \equiv \neg p \land \neg q$  $(p \land q) \equiv \neg p \lor \neg q$ olication  $\supset q \equiv \neg p \lor q$  $\exists x P(x) \equiv \forall x \neg P(x)$ uantification  $\forall x P(x) \equiv \exists x \neg P(x)$ Chapter 16: Logic Programming 28

## From Predicates to Horn Clauses

- Six-step procedure that will, whenever possible, translate a predicate p into a Horn clause.
  - Eliminate implications from p, using the implication property.
  - Move negation inward in p, using the deMorgan and quantification properties, so that only individual terms are negated.
  - Eliminate existential quantifiers from p, using a technique called *skolemization*. Here, the existentially quantified variable is replaced by a unique constant.
    - For example, the expression ∃xP(x) is replaced by P(c), where c is an arbitrarily chosen constant in the domain of x. Chapter 16: Logic Programming 29

# From Predicates to Horn Clauses

- 4. Move all universal quantifiers to the beginning of p; as long as there are no naming conflicts, this step does not change the meaning of p. Assuming that all variables are universally quantified, we can drop the quantifiers without changing the meaning of the predicates.
- s. Use the distributive, associative, and commutative properties to convert p to conjunctive normal form. In this form, the conjunction and disjunction operators are nested no more than two level deep, with conjunctions at the highest level.

Chapter 16: Logic Programming

| From Predicates to Horn<br>Clauses  |
|---|
| 6. Convert the embedded disjunctions to implications,<br>using the implication property. If each of these<br>implications has a single term on its right, then each<br>can be rewritten as a series of Horn clauses<br>equivalent to p. |
| • Example:  |
| <pre>∀x(¬literate(x)⊃(¬writes(x)∧¬∃y(reads(x,y)∧book(y))))</pre> Example:   |
| $\forall x(literate(x) \supset reads(x) \lor writes(x))$  |

Chapter 16: Logic Programming